PART FOUR

Apply Measures

Part 2 of this Guide describes how to select the measures that best address the issues of a particular project. This part of the Guide explains how to *Apply Measures* to gain insight into the project issues. It provides detailed guidance on generating measurement indicators. Three types of analysis are described: estimation, feasibility, and performance. Measurement is useful only when it provides information that helps to make objective and informed decisions about the project issues.

This part of the Guide is organized into four chapters:

- **Chapter 1, *Apply Measures* Overview,** summarizes the tasks of collecting, analyzing, and reporting measurement data and information.
- **Chapter 2, *Collect and Process Data,*** describes accessing and verifying measurement data prior to analysis.
- **Chapter 3, *Analyze Issues,*** explains the analysis of measurement data to create estimates, to determine feasibility of plans, and to assess project performance.
- **Chapter 4, *Make Recommendations*,** explains how analysis results are used to develop recommendations to address project issues.

[This page intentionally left blank.]

1

Apply Measures Overview

The measurement process must respond quickly to the information needs of project managers. Typical questions asked by project managers include:

- Can I trust the data?
- Is there really a problem?
- How big is the risk?
- What is the scope of the problem?
- What is causing the problem?
- Are there related risks?
- What should I expect to happen?
- What are my alternatives?
- What is the recommended course of action?
- When can I expect to see the results?

The *Apply Measures* activity should generate answers to these questions.

When analysis follows a defined and repeatable procedure, the results are more likely to be credible and complete, and the project manager will have a higher degree of confidence in them. PSM presents the analysis activity from three perspectives: 1) a model of relationships among issues that helps to guide the analysis, 2) indicators that present measurement information about issues for analysis, and 3) the types of analyses conducted. The three types of analysis discussed include estimation, feasibility, and performance.

The *Apply Measures* activity converts measurement data into information that relates directly to the project issues. Figure 4-1 shows the major tasks for collecting, processing, and analyzing measurement data that provide feedback for effective decision making.

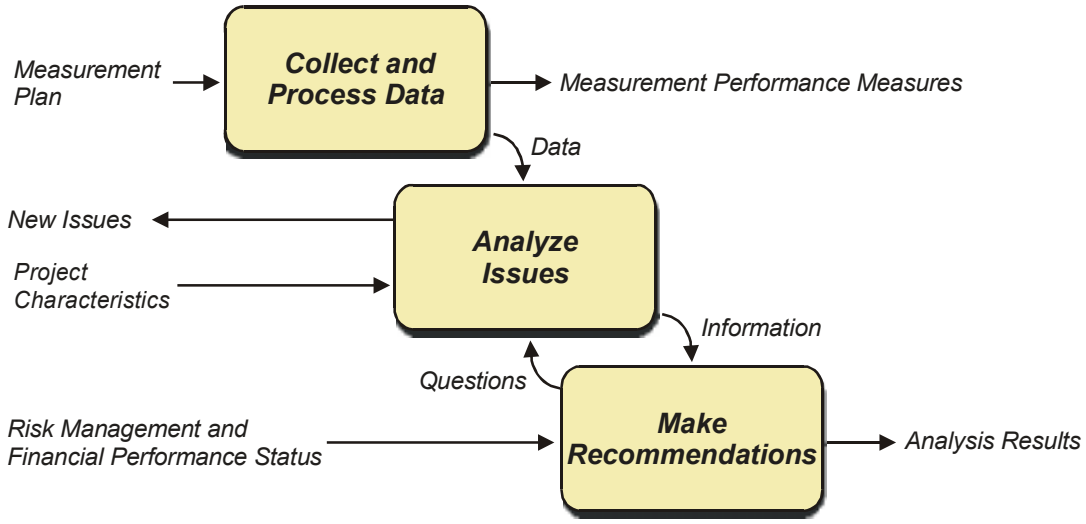

Figure 4-1. *Apply Measures* Activity

The measurement plan that was created during the tailoring activity identifies project issues and specifies the measurement data to be collected. In the first task of the *Apply Measures* activity, measurement data is collected and prepared for analysis. This task, entitled *Collect and Process Data*, is discussed in Chapter 2. The next task, *Analyze Issues*, is described in Chapter 3. In this task, previously collected data is converted into information that can be used by the project team to make decisions. The transformation of data to information occurs through a number of systematic analysis steps. The final task, *Make Recommendations*, is discussed in Chapter 4. Information from this task helps project managers make decisions and take corrective action with respect to project issues. The *Apply Measures* activity is repeated periodically throughout the project life cycle.

2

Collect and Process Data

Collecting and understanding the data is the first task in the *Apply Measures* activity. Valid data is the foundation of any measurement process. Some of the concerns associated with data collection are the sources, reporting frequency, format, normalization and aggregation, conventions, and verification.

The collected data should closely reflect the nature of the product or process being measured. Be sure to include all contractors and subcontractors in the data collection effort. Mature organizations are likely to provide more data at greater levels of detail than less mature organizations.

Figure 4-2 shows the three key steps involved in collecting and processing data.

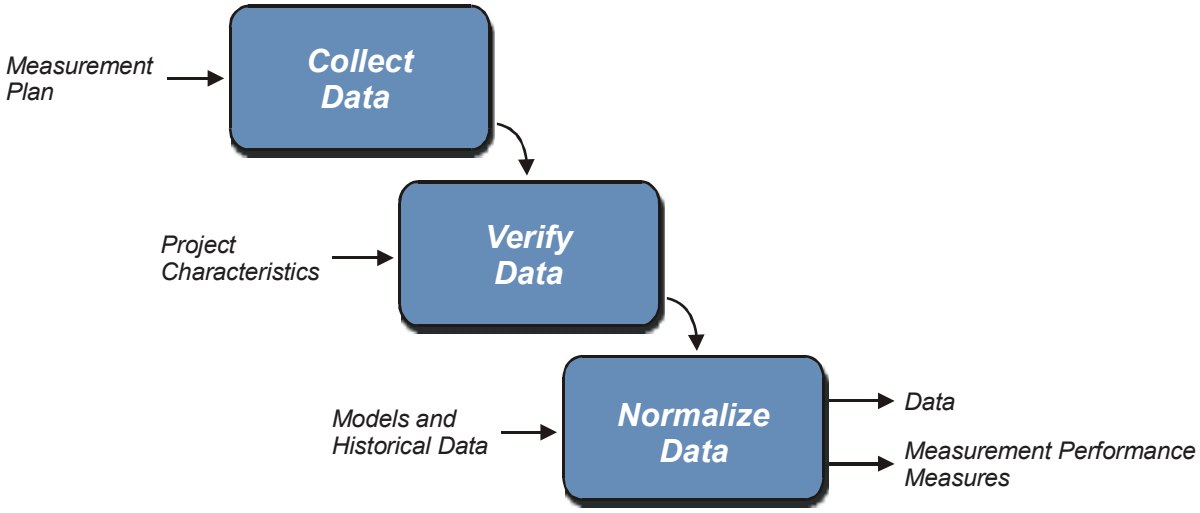

Figure 4-2. ***Collect and Process Data***

Measurement data is generated from activities and products throughout the life cycle. Data may be collected via forms or automated tools. Once data is collected, it needs to be verified. Verified data may then need to be normalized before it is used for analysis purposes.

The following sections discuss each of these steps in detail.

2.1 *Collect Data*

Data definitions that were developed during the *Tailor Measures* activity and documented in the measurement plan provide detailed specifications for each data item. The plan also defines certain

information such as what data will be accessed, when it will be available, what format it will be in, and what software tool or source it will come from. Confirm that actual data matches the measurement plan's specifications.

Personnel responsible for generating measurement data should provide data to the measurement analyst responsible for collecting and processing it. Figure 4-3 illustrates how data access may occur. Common data access mechanisms include:

- **Direct, shared access -** The measurement analyst has access to the project data sources or to a project-level measurement repository.
- **Electronic copies of files and/or databases -** The project team regularly makes copies of files or databases in their native format and provides them to the measurement analyst. The measurement analyst must have the tools to read and interpret those files.
- **Data exports -** Each reporting period, the project team exports data from each data source into a standard format, such as an ASCII delimited text file. The measurement analyst imports the data into an available file system or measurement tool.
- **Hardcopy -** Each reporting period, the project team provides printed output showing the measurement data generated for that period. The measurement analyst must then enter the data or perform the analysis manually.

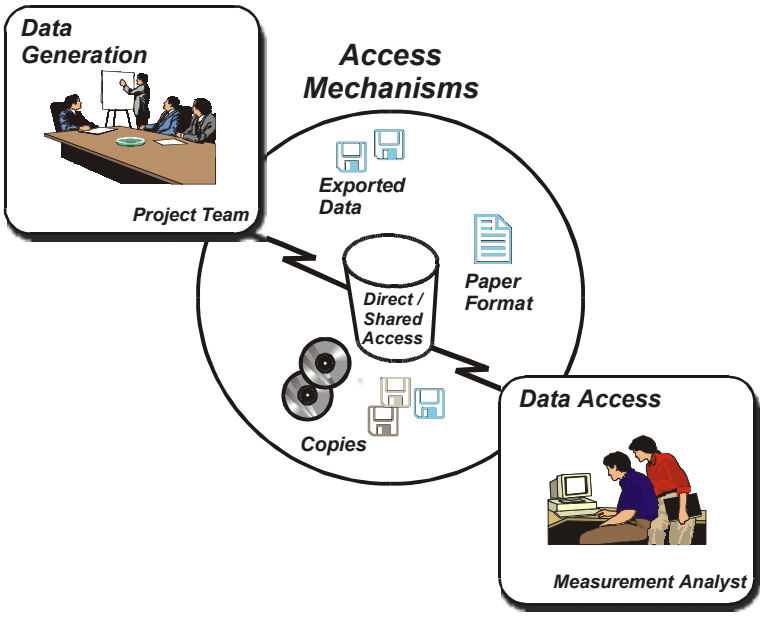

Figure 4-3. Common Data Access Mechanisms

Direct access to the project's data repository or sources is the preferred access mechanism. This "shared data" approach has a number of benefits. Direct access provides the measurement analyst with timely access to the same data as the project team. Direct access also ensures a level of detail that allows independent identification and analysis of problems. This access approach also reduces the cost and impact on the project team. Most data users have read-only privileges and sensitive data can be protected with password and authorization techniques.

The lag time between collecting, analyzing, and reporting data should be as short as possible. On-line access to the project databases facilitates data timeliness. Without a shared data strategy, the lag time between data generation and data receipt must be factored into the measurement analyst's schedule.

The project team may collect data more frequently than the measurement analyst analyzes it. The most common collection intervals are monthly for requirements analysis, design, and implementation, and weekly for integration and test activities. However, on small projects with short cycle times or for projects using a rapid incremental development approach, weekly reporting of data may be necessary. In addition, as a project approaches key milestone decisions, more frequent data collection may be needed. On the other hand, product quality data (other than defects) is often collected only at the end of a phase or upon completion of a major product component.

2.2 *Verify Data*

Useful measurement results depend on good data. Data verification must consider both the accuracy of the data as it is recorded and the fidelity with which it is transmitted. All data should be identified with its collection date and its source, to align data with project events and to correlate data from different sources. Configuration management of data delivery versions and dates should be implemented for electronic data sets. Check this audit trail periodically to assess the integrity of the data collection process.

Communicating clear definitions of data items helps ensure consistent data. Conformance to obvious terms must be verified; for example, the average number of hours worked per staff month varies from organization to organization.

Data verification is complicated by the fact that some of the assumptions underlying the measurement process can change during the project. Aggregation structures, product components, life-cycle processes, and even measurement definitions may be updated or revised as the project evolves. Sometimes, estimates and actuals are measured differently. Definitions, assumptions, and aggregation rules must be clearly understood. Make sure the data provided matches the current measurement specifications.

Figure 4-4 contains examples of typical data verification questions to consider.

Data Verification Checklist	
1	***Data Currency*** • Does the data relate to the project activities currently underway? • Does received data match the schedule?
2	***Data Aggregation Structures and Attributes*** • Are values in the fields for aggregating data records consistent across records (for example: defect classifications, component identifiers of configuration items or units, project activities, work unit packages, and cost accounts? • Are values in the fields for aggregating data records consistent across project teams or organizations?
3	***Units Of Measure*** • Are the same units of measure being used across all project teams or organizations, for example, hours versus days and lines of code (SLOC) versus thousands of lines of code (KSLOC)?
4	***Data Item Contents*** • Are any data values outside the acceptable ranges? • Is the format of any data item's values incorrect, for example, data type and decimal positions?
5	***Data Completeness*** • Is needed measurement data provided for each issue? • Are data items missing within measurement data? • Is project-context data delivered? • Is this the data agreed upon?
6	***Changes To Existing Data*** • Have plan values changed unexpectedly, for example, planned start and end dates, planned cost/staffing/effort, and planned components completed/tested? • Do changes to plan values represent a major replan? • Have any actual values changed unexpectedly, for example, actual start and end date for activities already completed and actual cost/staffing/effort for prior periods?
7	***Does The Data Look Too Regular?***

Figure 4-4. Typical Data Verification Questions

Recognize that even accurate and verified engineering data may be *noisy*. Software and systems engineering are human-intensive activities; things seldom go exactly as planned. Because performance varies from week to week, the analyst should be wary of *actual* data that exactly matches the *plan*.

Any data concerns or inconsistencies should be resolved through communication with the project team. Missing data, significant changes in values, or changes in the data structure should always be discussed with the project team for complete understanding.

2.3 *Normalize Data*

Before data can be analyzed, it may need to be normalized. *Normalization* is the process of converting data into a common unit of measure and/or aggregation level so it can be compared or combined with other data. Examples include:

- Converting one team's effort from hours to months so that is can be compared to another team's monthly effort data
- Converting a subcontractor's life-cycle activities to the prime contractor's different set of activities to combine and analyze effort by activity

Normalization must be done carefully; rules and procedures should be documented.

Insight into project issues begins with the task of collecting and processing data. The availability, timeliness, consistency, completeness, and accuracy of the data will determine the value of the resulting information.

[This page intentionally left blank.]

3

Analyze Issues

During the *Analyze Issues* task, data is converted into information that is used by managers to make decisions relative to project-specific issues. Data is transformed into information through a systematic series of analysis steps. This task is repeated throughout the project's life.

The focus of analysis changes as the project evolves. Figure 4-5 shows three types of analyses. Specific analysis techniques depend on data available and the information needs of the project. As shown in Figure 4-5, each analysis type has its own inputs, and produces different types of results.

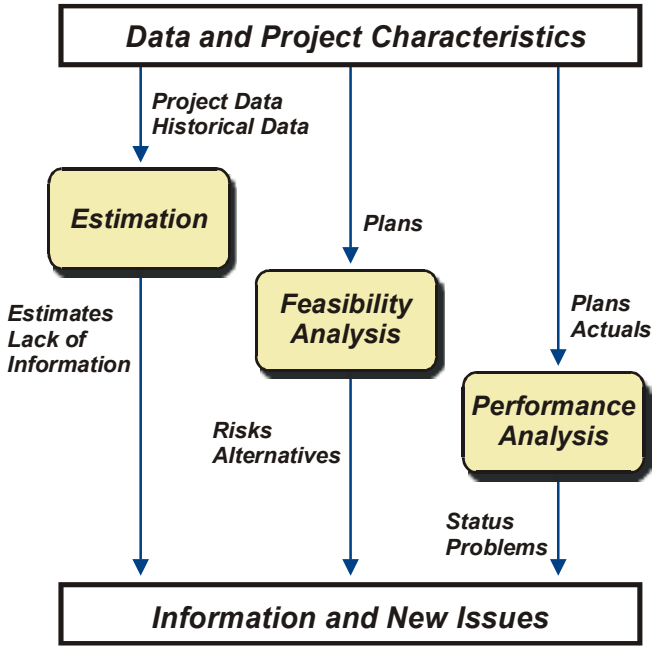

Figure 4-5. Three Types of Issue Analysis

Early in the project life cycle, the focus is on *Estimation* to support project planning. *Estimation* establishes target values for size, effort, and schedule. *Estimation* usually starts with historical data and assumptions about the project's process and products. *Estimation* also identifies uncertainties that feed back into the issue identification activity. Estimation should be conducted during the initial planning phase and during all subsequent replans.

As plans near completion, the focus shifts to *Feasibility Analysis*. This type of analysis determines whether project plans and targets are technically realistic and achievable. *Feasibility Analysis* uses historical data, experience, and consistency checks to evaluate the project plans. Risks identified during this analysis should

be entered into the project's risk management process. *Feasibility Analysis* should be conducted during the initial planning phase and during all subsequent replans.

Once the project has begun, *Performance Analysis* becomes the major concern. *Performance Analysis* determines whether the project is meeting defined plans, assumptions, and targets. Inputs include plan and actual performance data. *Performance Analysis* is designed to identify risks, problems, and corrective actions. *Performance Analysis* should be conducted on a regular basis.

Analysis tasks are, by their nature, investigative. Each issue may require applying a different set of analysis techniques in order to isolate, understand, and facilitate resolution of a problem. As project problems, risks, and information change, the types of analyses performed and the indicators generated must be revised.

3.1 General Analysis Concepts

Indicators are the basic building blocks of measurement analysis. This section defines measurement indicators and describes how to use them to address project issues. In the PSM process, an *indicator* is defined as *a measure or combination of measures that provide insight into a project issue or concept.* An indicator is often represented as a graph or a table. An important issue may be addressed with several indicators and in many cases the indicators are based on different measures.

The PSM measurement approach emphasizes the collection of data at a low level of aggregation. Many different indicators can be constructed from this data. The analyst can combine and present measurement data in many different ways, depending on what the project situation requires. It allows greater flexibility in analyzing critical issues and in adapting to new issues as they arise. A measurement process that is based only on the periodic delivery of pre-defined graphs does not have this flexibility.

3.1.1 Basic Indicator Concepts

The design of an indicator is important. In the PSM process, measurement data is selected to address a specific issue. To be effective, an indictor must present data in a format that facilitates the clear interpretation of data and the derivation of information to manage the issue. Indicators should be designed to:

- Provide information about the intended issue
- Support the type of analysis that is needed (*Estimation, Feasibility*, or *Performance*)
- Provide the appropriate level of detail
- Provide timely information for making decisions and taking actions

The PSM concept of indicators provides a systematic method for examining measurement data. In most cases, insight into an issue cannot be obtained by collecting only actual performance data. Actual data must be compared with an *expectation* of what it should be. That expectation may or may not be explicitly stated prior to the start of the measurement process. It may be a rule of thumb such as, "error rates usually go down as testing progresses." Criteria are needed to decide whether or not the difference between actual data and expected data is significant. Thus, the basic PSM measurement indicators consist of three parts:

- **An actual value of a measure or combination of measures** - Examples include hours of effort expended, or lines of code produced to date.

- **Expected value of a measure or combination of measures** - This is a planned value, quantitative objective, baseline, or historical value such as planned milestone dates, target level of reliability, or required productivity.

- **Decision criteria** - These include rules of thumb and statistical techniques used to assess the difference (often called variance) between planned (expected) and actual (measured) values.

Figure 4-6 provides an example of an indicator that contains these three elements of the indicator. The actual values show the cumulative number of components that have completed design to date. The expected values are represented by the plan line, which shows the amount of design that should be accomplished at any time during the design phase. The variance is the gap between the actual and the last plan values. The decision criteria might be that a variance of more than 10 percent requires further analysis. For example, for December 1999, 67 percent of the units expected to be complete were not complete, which exceeds the 10 percent threshold.

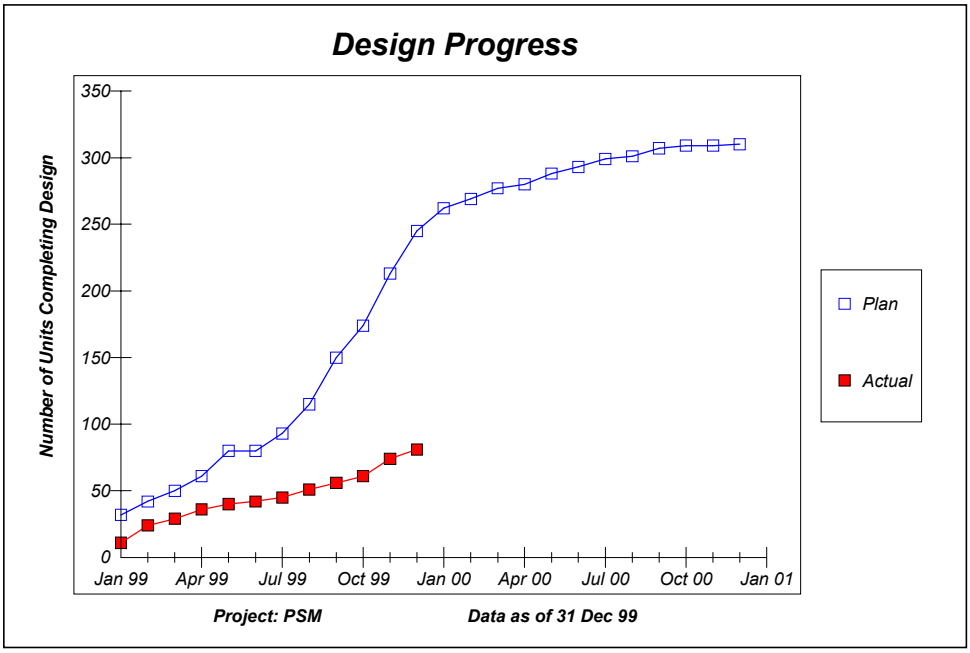

Figure 4-6. Example of an Indicator

3.1.2 Structured Analysis Model

Indicators help provide insight into project issues. It is important to note that issues are not independent of one another. In order to define the relationships between individual issues, PSM uses a "structured analysis model" shown in Figure 4-7. This model illustrates the typical relationships among the PSM common issue areas and sets the stage for defining appropriate measurement indicators for analysis activities.

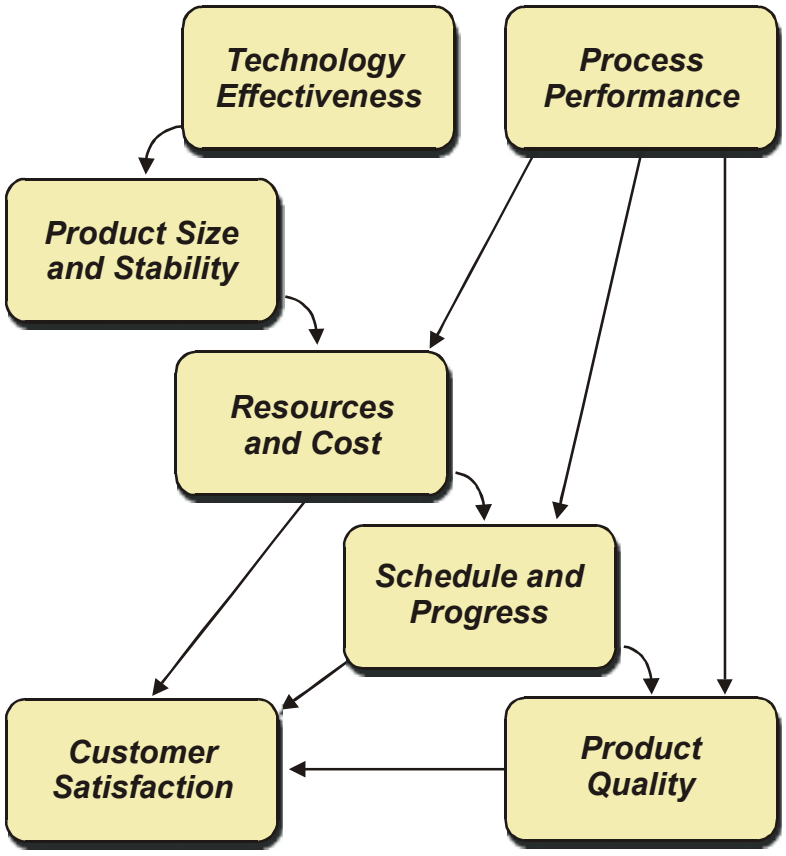

Figure 4-7. Issue Analysis Model Showing the Relationships Among Common Issue Areas

Each of the arrows in the figure represents a relationship. For example, as Product Size and Stability changes, so does Resources and Cost. Each of the common issue areas is related to its neighbor in the structured analysis model. A more detailed version of the analysis model appears in Figure 4-8. This figure shows the PSM common issue areas divided into their respective measurement categories. Use the structured analysis model to understand the relationship between common issue areas and to select measures and indicators that provide information for issues of concern.

Upstream issues often provide an early indication that a problem may occur in one of the downstream issues. Upstream issues also explain the cause of a current problem. For example, if schedule and progress is a high-priority issue, effort or size might be investigated as the cause of a schedule problem. Collecting effort and size measures can either serve as a potential warning of schedule slippages or identify the cause of the slippage.

Downstream issues may indicate how a management decision will affect a current issue. For example, a management decision to change the product size may affect the issues of effort and schedule. An increase in size may require more people and more time, resulting in increased cost and schedule slippage.

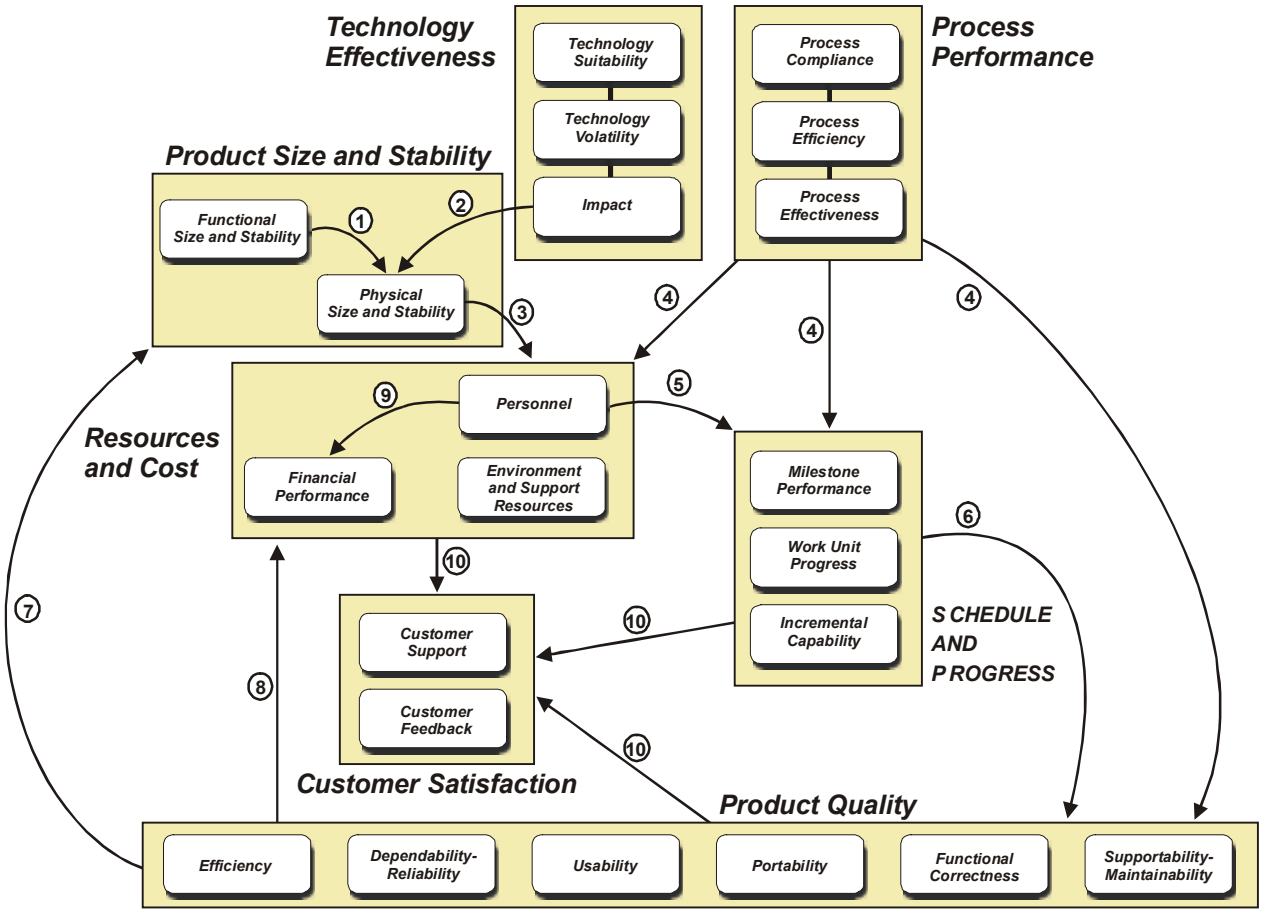

Figure 4-8. Model of Common Software Issue Relationships

The following relationships correspond to the numbered circles in the figure:

1. Functional size represents the amount of functionality that the system must provide. This is usually determined by the requirements. Functional size is a primary determinant of physical size (the amount of product that must be developed or maintained).

2. The impact and volatility of any new technology also influences product size. Most innovative technical approaches attempt to minimize the amount of new product that must be implemented for a given function. Examples of technical approaches include COTS, common architectures, and reusable components. If the effectiveness of the approach does not yield all of the desired benefit, more of the system must be developed than planned.

3. Increases in product size usually result in the need for additional personnel.

4. Process performance affects the overall need for personnel resources, and influences development schedules and product quality. A more capable organization performs better, assuming that other factors are constant.

5. Adding more personnel impacts schedule and progress. If personnel are added early in the project and if the appropriate training and communications are in place, the schedule may be shortened. If personnel

are added later, schedule delays may occur because it is difficult to add unplanned staff to an ongoing project. Schedule shortfalls are associated with milestone slips, delays in completing planned life-cycle activities and products, and downsized build and release requirements.

6. Schedule shortfalls can cause product quality problems, including defects in the product. In order to meet the schedule, test efforts may be curtailed or problems may not be corrected. In general, higher system complexities make maintenance more difficult, requiring more time to fix errors.

7. Latent quality problems represent rework that requires additional effort to make current or future releases acceptable to the user. The project manager will usually make a delivery decision based on the number of open problems. High-priority problems may be fixed, while others may be deferred to the operations and maintenance phase.

8. Rework increases the effort required to complete the project as originally specified. Projects are often forced to modify or eliminate some mission requirements to stay within cost and schedule constraints.

9. For software-intensive systems, personnel effort (including rework) is usually the primary determinant of project cost. Cost control can be achieved only by controlling other upstream factors.

10. Resources, schedule, and product quality all impact customer satisfaction.

Example Using the Structured Analysis Model

The following is an example of how the structured analysis model can be used. Figure 4-9 is an indicator that shows how the estimated size of a software component has changed since the initial estimate.

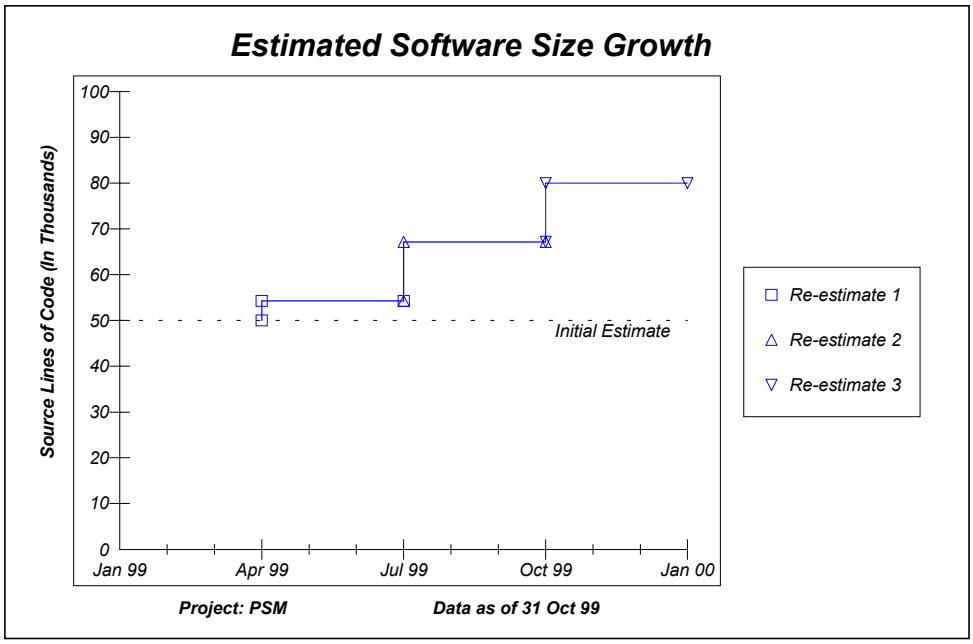

Figure 4-9. Size Growth Indicator

Significant size growth has occurred on this project. This is a leading indicator of a potential need for additional development effort, which will impact project cost and schedule.

Figure 4-10 is an effort indicator that shows the variance between planned and actual effort. Although the applied effort is tracking close to plan, the size growth indicator (Figure 4-9) implies that the planned effort will be insufficient. The planned effort in Figure 4-10 probably has not been revised to reflect the re-estimate of size depicted in Figure 4-9. This is critical, given its downstream impact on cost and schedule.

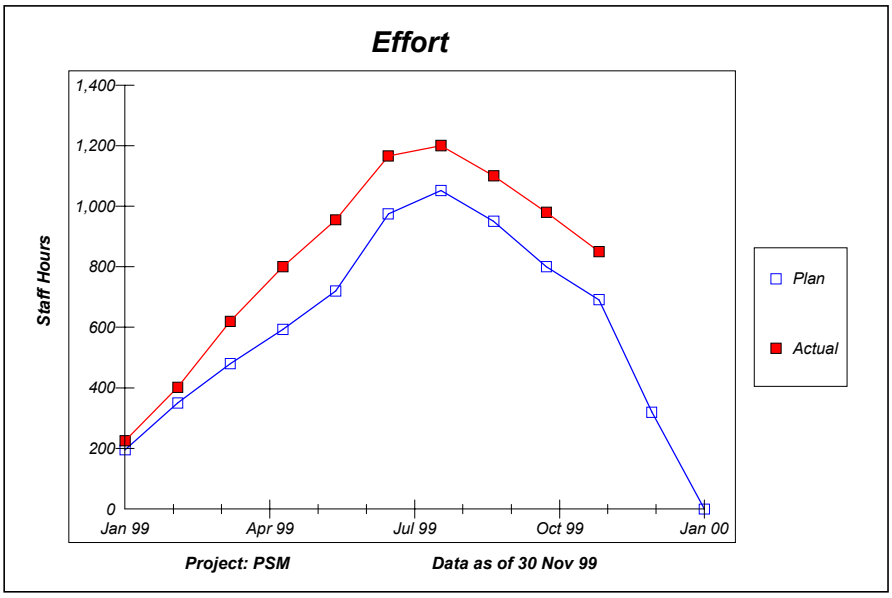

Figure 4-10. Effort Expenditure Indicator

The size growth indicator in Figure 4-9 can also be used to help quantify the amount of expected effort growth. Using the "estimator" depicted in Figure 4-11, the amount of size growth from Figure 4-9 is used to define a new plan for total project effort.

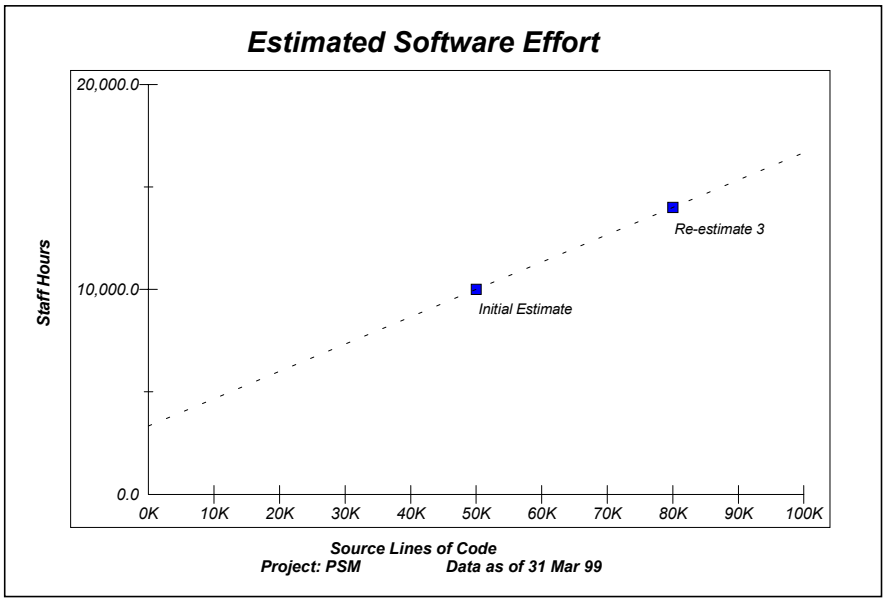

Figure 4-11. Estimator Generated From Measurement Data

Use of the structured analysis model and the generation of indicators are dynamic. The analysis model is applicable to *Estimation*, *Feasibility Analysis*, and *Performance Analysis*. Measurement indicators change with the analysis process to answer different (but related) questions, and to provide different views of measurement results.

3.2 *Estimation*

Estimation is a type of measurement analysis used to establish target values or numerical expectations for subsequent project activities, based on currently available data. Estimation typically produces projections of the product size, effort, and schedule required to complete the project. Estimation may also produce projections of product quality. These estimates form the basis for initial project plans and subsequent replans. Some projects also estimate system performance targets such as memory utilization and throughput. It is important to complete estimates for key project measures (such as size and effort) at several points during the life-cycle process.

Estimation is the first type of analysis on most projects. The initial round of estimation often occurs before the project team is selected. These initial estimates support the cost-benefit analysis used to justify the project and to establish its overall funding and schedule commitments. However, early estimates are often imprecise and must be updated throughout the project life cycle.

Poor estimates and misconceptions about the estimating process often contribute to failed projects. A poor estimate leads to infeasible plans. Implementing such plans results in missed deadlines, inadequate performance, and poor quality. Poor estimation can be attributed to a number of factors:

- Lack of historical data on which to base estimates
- Lack of estimating experience
- Lack of a systematic estimation process, sound techniques, or models suited to the project's needs
- Failure to include essential project activities and products within the scope of the estimates
- Unrealistic expectations or assumptions
- Failure to recognize and address the uncertainty inherent in project estimates

The guidance in this chapter cannot substitute for valid historical data or estimation experience. However, it does explain what data, models, and tools are needed to estimate systematically. The PSM guidance includes the structured analysis model, suggested measurement indicators, and a basic estimation process that help ensure that comprehensive and realistic estimates are produced.

Using the Analysis Model

The analysis model described in section 3.1.2 can be used to support estimation. The elements of Technical Effectiveness and Process Performance influence the relationships among the other elements. As size and functionality increase, the effort to implement that functionality increases. On the other hand, as effort increases, *for a given amount of functionality (and rework),* the schedule required to complete the work decreases, although not proportionately. Many studies have shown that this is a costly way to reduce schedule.

Any element in the analysis model that is upstream from a given element can be used as a basis for estimation. For example, either functional size or product size can be used to estimate effort. The strongest relationships are found between elements that are close together in the model. However, in most cases, a project's estimation information is neither complete nor entirely accurate. For example, if the final component size in lines of code were known, a good prediction of effort could be made. However, this is often impossible early in the life cycle, when it is usually hard to get an accurate determination of lines of code. Functional size measures such as requirements or function points may give better results for early estimation purposes.

Estimators

Estimation uses one measure to predict the value of another. Estimation uses a special type of indicator called estimators and adjustments referred to as performance factors. *Estimators* show the predictive relationship between two measures. For example, an estimate of the amount of code can be used to predict the amount of effort required to produce it. These estimating relationships are often domain specific. *Performance factors* adjust the relationship to account for the specific project situation. Examples of common predictive *estimators* include:

- Functional size to predict product size
- Functional size to predict effort
- Product size to predict effort
- Effort to predict schedule
- Effort to predict cost
- Product size to predict the number of problems (quality)

Examples of performance factors include:

- Effectiveness of COTS technology
- Historical productivity
- Staffing capability
- Historical error rate
- Stability of requirements

Estimation models vary in their estimating relationships and performance factors. For example, some models assume linear relationships between size and effort, while others assume non-linear relationships.

Figure 4-12 shows a simple estimator for project effort based on product size. The solid line shows the estimating relationship between size and effort. As the amount of software increases, the amount of effort required to produce the software also increases. The plotted points represent data from past projects. The dashed lines define a region around the estimating relationship into which 95 percent of the data falls. For any given product size, there is a 95 percent chance of actual effort falling somewhere in that region.

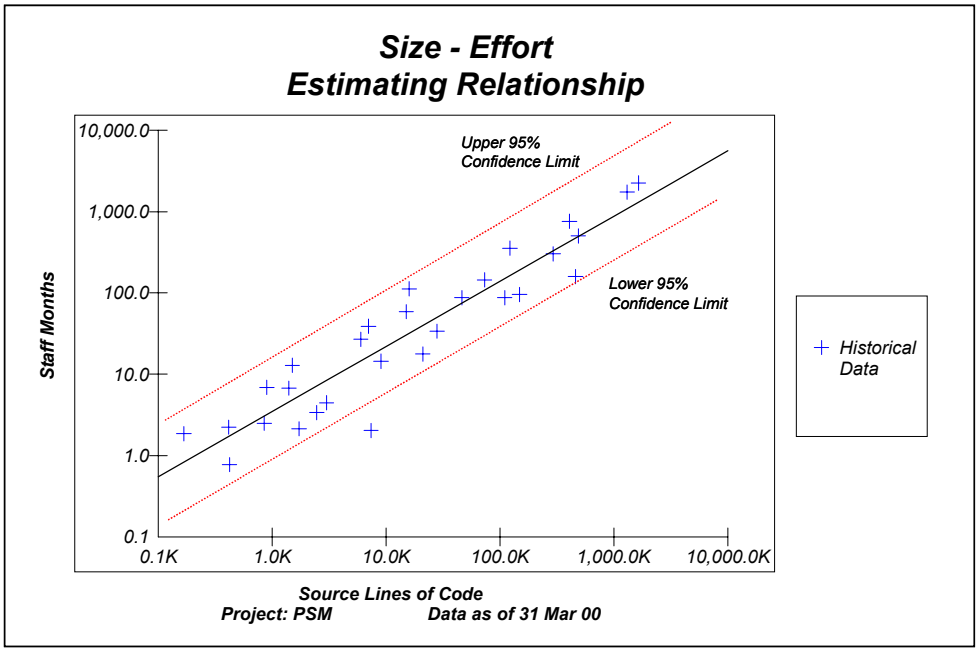

Figure 4-12. Estimator for Effort

Note that not all of the points lie on the center line. This indicates that the relationship between size and effort is not *deterministic*, meaning that even if the precise value of size is known, the precise value of effort cannot be determined. Unfortunately, many estimation models only produce single point estimates. Obtaining a probabilistic statement of the estimate, such as can be derived from Figure 4-12, may be more realistic. This figure shows that the 95 percent confidence interval for the effort required to develop 100,000 lines of code ranges from about 10 to 250 staff months. This indicator provides the uncertainty level assumed in the estimation process. In this example, there is a very wide confidence interval. Additional parameters should be considered in order to reduce the uncertainty. Understand the estimate's level of confidence as well as the estimated value itself.

Estimation Task

The total system cost (or program cost) is the sum of the costs for all elements of the system (hardware, software, systems engineering, and program management). This includes direct purchases of software and hardware and all costs for designing, developing, or procuring these components or the system as a whole. Regardless of the project element to be estimated, estimation typically involves four basic steps (as shown in Figure 4-13):

- ***Select Approach*** - Select an estimation approach for the element (e.g., hardware, software, or systems engineering). The approach may use a mathematical model or a technique based on simple estimating relationships.

- ***Map and Calibrate*** - Map the approach to the project's sequence of life-cycle activities and products, and calibrate associated estimation models with historical data from the organization.

- ***Compute Estimate*** - Compute estimates of size, effort, schedule, and quality using the approach or model.

- ***Evaluate Estimate*** - Compare the results with project constraints and assumptions to evaluate the estimate. If the estimate does not satisfy the project constraints, then make appropriate adjustments and redo the estimate.

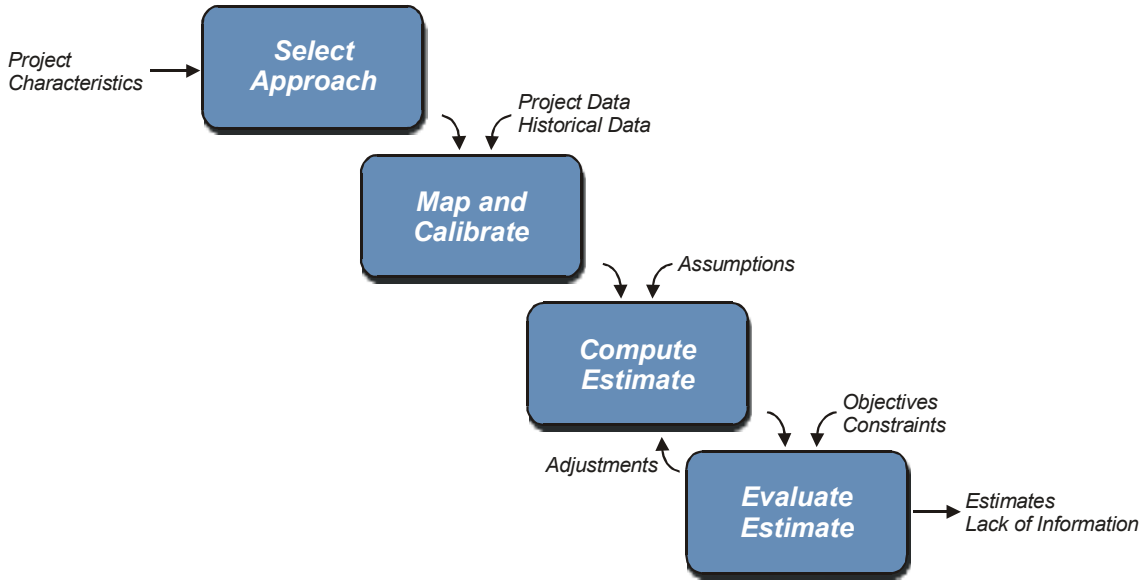

Figure 4-13. *Estimation* Task

Estimates are the foundation for more detailed planning. They should be documented in the project management plan. Re-estimates should be performed periodically throughout the project life cycle, at major milestones and when changes or project constraints require a new plan. As the project progresses and actual performance data becomes available, the accuracy of estimates increases.

Hardware Cost Estimation Techniques

Later sections describe how to apply the estimation activity using software examples. Key concerns and alternative techniques are discussed for each step. The same steps apply when estimating hardware-engineering costs. While the general process of estimation is similar, the specific models, tools, and data used to estimate software and different types of hardware vary. Parametric models for hardware engineering use physical characteristics such as spatial dimensions, number of ports, or weight instead of lines of code. Cost estimates for hardware components may also be based on a bill of material for components to be purchased.

System Engineering Cost Estimation Techniques

The systems engineering activities of a project may include system requirements development, design, integration, testing, deployment, and maintenance and operations activities. Systems engineering and program management costs are often computed as a percentage of hardware and software costs. The specific allocation for these activities depends on their anticipated difficulty, personnel skill levels, and other factors similar to those discussed below for software.

3.2.1 *Select Approach*

The first step in the estimation task is to select an appropriate estimation approach. For software, many different estimation methods and tools are available. Comparable techniques support estimation of hardware and systems engineering activities. The following sections describe estimation approaches, provide selection criteria, and explain how to use these approaches.

Types of Estimation Approaches

Estimates are usually developed using one or more models or techniques. A model is an idealized representation of real-world relationships. It may be a complex mathematical formula, a simple arithmetical expression, a set of rules, or a list of descriptive statements. Regardless of the model's sophistication, the quality of its estimates is no better than the assumptions and the data entered into the model. Four major types of estimating models are described below.

Parametric Models

Parametric models assume that one or more mathematical relationships exist among size, effort, schedule, and quality. These models also assume that the relationships are affected by measurable performance factors (also called parameters). These relationships are based on theoretical reasoning or analysis of historical data.

Parametric models generally take the form:

$$\text{Effort} = A \cdot (\text{Size})^{B} \cdot C$$

The model's output is estimated effort expressed in hours or staff months. The most important input into the model is product size, which represents the amount of work to be accomplished. The size parameter is the most significant predictor of effort. Parameter A is constant. Parameter B is an aggregation of performance factors that affect the model output (effort) in a non-linear manner. These non-linear performance factors adjust large size parameters more dramatically than small size parameters. This causes the factors to have more influence on larger software projects than on smaller ones. Parameter C is an aggregation of performance factors that affect effort in a linear manner. These linear factors have a directly proportional influence on effort regardless of product size.

Performance factors generally fall into two categories: process factors and product factors. Process factors are based on the characteristics of the project's life-cycle process, including the tools used and the skill level of project personnel. Product factors include attributes such as the operational environment, required reliability, and application complexity. The values of these parameters are selected to raise or lower productivity to more accurately reflect expected project conditions. While estimation models may provide 15 to 100 adjustment factors, most organizations find that only a few factors strongly affect their performance.

For the best results, parametric models must be calibrated to data from the local development environment. Calibration usually adjusts only the constants in the model (such as the constant A described in the general model above). While calibrating all of the parameters takes a significant amount of data, only a few data points may be required to calibrate a model's constants to local development conditions. Most estimation tools recommend calibration.

The Constructive Cost Model (COCOMO II) (Boehm, et. al., 2000) is an example of a parametric model for estimating effort and schedule from product size, either in source lines of code or function points. After size is entered, twenty-two performance factors are applied to yield an effort estimate in staff months. For example, one of these factors is applications experience, a linear performance factor. A high level of

experience reduces the estimated effort by 23 percent, while a low level of experience increases it by 22 percent.

Most parametric models assume that schedule is a function of effort. Most models subdivide schedule and effort estimates by major activity or phase. The activities and phases assumed by the models may not match the project plans, necessitating adjustments. Few parametric models support both effort/schedule estimation and quality estimation.

There are numerous estimation tools for parametric effort and schedule models. Some are free, while others are quite expensive. The level of vendor support also varies substantially.

Activity-Based Models

Activity-based estimation is sometimes referred to as activity-based costing or bottom-up estimating. The activity-based approach relies on collecting engineering estimates of effort and schedule for all product components and tasks in a project. Size estimates may also be collected. This information is used to estimate each individual activity for each product. The estimates are then aggregated to produce a project-level estimate.

For example, effort to design each lower-level component may be estimated and then added together for an estimate of system design effort. Some bottom-up approaches use project-level characteristics to inflate or deflate the estimate for product, project, and technical risks. The activity-based estimation approach requires detailed knowledge of the product and process.

When developing an activity-based estimate, be sure to include the effort to produce all products and documentation, regardless of whether they are used internally or delivered to the customer.

Analogy

The analogy approach is based on a comparison of the characteristics of the proposed system with other previously completed systems. Data from similar or analogous systems is the basis for the proposed system's estimates. Differences between the systems are identified and appropriate changes are applied to adjust the size, effort, schedule, and quality to fit the new situation. While estimates based on analogy can be generated from just one similar project, getting an accurate estimate requires a detailed understanding of both the analogous and estimated project.

Simple Estimating Relationships

Another estimation approach is a simplification of the parametric modeling approach. Simple estimating relationships based on local historical data are used instead of a comprehensive mathematical model. Examples of simple estimating relationships include effort as a function of product size and productivity, and error rate as a function of size. For example, productivity can be used to estimate the effort required to develop a new product within the same organization:

Effort = Product Size * Productivity

where Productivity = Staff Months/ Product Size

The productivity number is determined from past projects. The size input is an estimate of the new product's size. The result is estimated staff months of effort. This estimate is then used with data on the percentage of effort spent in each phase of the development (collected from past projects) to estimate the amount of effort for each phase of the new project. For example, assume that the effort estimate for the new project was 50 staff months and the percentage of effort spent in the design phase from previous projects was 33 percent. In this case, the estimated design effort required for the new project would be 16.5 staff months.

Estimating relationships generally do not apply outside of the organization and domain that provided the data. This is because a simple relationship (like productivity) assumes that all of the performance factors that affect effort are similar among past and future projects. If these factors are significantly different, then a parametric model should be used.

Factors in Selecting an Estimation Approach

All of these estimating methods yield good results under the right circumstances. When selecting an estimation approach, consider how closely its assumptions match the project and whether the data required by the approach is available from a reliable source. Some specific factors to consider in selecting an estimation approach include:

- Whether the activities covered by the model or approach match the planned activities for the project in question
- Validity of the approach at different levels of project aggregation (some approaches/models are better applied at the system level than at the component level)
- Quantity, quality, and type of historical data on which the model is based
- Level of understanding of the system being estimated
- Availability of local historical data to calibrate the model or approach
- Applicability to the type of product being developed, such as COTS and reused software
- Availability of actual "to date" data from the project to produce "estimates to complete"
- Ability to provide reasonably accurate values for the parameters that must be provided as input to the model
- The price, documentation, and support of the estimation tool
- The mathematical difficulty of implementing the approach and understanding its results

Figure 4-14 summarizes three of the most important factors for the four basic estimation approaches.

Estimation Approach	Understanding Assumed	Historical Data Required	Mathematical Complexity
Parametric Models	General descriptive information	Multiple projects to develop the model; none to use the model	Complex statistical techniques
Activity-Based Estimates	Detailed process and product information	Very detailed data for a few projects	Arithmetic
Analogy	Detailed product information	At least one similar project	Arithmetic
Simple Estimating Relations	General descriptive information	Multiple projects	Simple statistical techniques

Figure 4-14. Key Considerations in Selecting an Estimation Approach

The four estimation approaches differ significantly in the assumed level of understanding of the intended product application. Most parametric models only require an estimate of size and ratings for the appropriate performance adjustment factors. Activity-based estimation requires a detailed understanding of both the product to be implemented and the process to be followed. Analogy requires detailed project knowledge to recognize specific differences between the proposed system and the system used in the estimate. Simple estimating relationships require minimal knowledge of the application.

All four estimation approaches require some historical data to produce meaningful results. Most popular parametric models are based on an analysis of extensive historical data from many organizations. However, the best results are obtained when the model has been calibrated with local data (see the section on Parametric Models, above). Use a parametric model without calibration only as a last resort. Activity-based estimation requires detailed data on the process and product. Typically, data should be available for several projects so that a probable range of performance can be estimated. Analogy requires data from at least one similar project. Simple estimating relationships require data from multiple projects within the organization developing the project.

The conceptual and mathematical difficulty of the four approaches varies considerably. This is an important consideration when trying to understand what that the estimate means. Parametric models generally involve complex mathematical formulas. Calibrating a parametric model may require knowledge of statistical techniques such as multiple and non-linear regression. Activity-based estimation and analogy only require basic arithmetic. However, product and process structures can be complex, and estimating each component may be arduous. Simple estimating relationships require only a limited knowledge of statistical concepts.

Using Estimation Techniques

Selecting and using an initial estimation approach does not satisfy the project's need for estimation information. Parametric models are often used early in the life cycle because they produce an estimate based on little input data. For most models, all that is needed is an approximate product size. Of course, estimates based on models that have not been calibrated with local data, or whose performance factors have not been adjusted to the realities of the project, frequently turn out to be wrong by an order of magnitude. Unfortunately, many project managers act on these early estimates as if they were certainties. Applying

multiple methods and re-estimating periodically throughout the project life cycle can reduce the level of uncertainty.

Confidence in an estimate increases when more than one approach is used, especially if not all assumptions of one method have been satisfied. For example, the results of a parametric model can be checked against results from a simple estimating relationship. Adjustments for the assumed effects of performance factors can result in productivity that is far beyond the organization's past level of performance and may produce an unrealistic estimate.

It is also important to re-estimate at various points throughout the project. Information increases over the life of the project, yielding increasingly accurate estimates. Later estimates should be based on actual project data collected to date. There are several ways in which new estimates can be developed. For example, a parametric model could be re-run with new size inputs that may include the actual results for some product components. Subtracting the budget and schedule expended to date yields the estimate to complete. Alternatively, the profile generated by the estimate can be compared with actual project performance. For example, if the project has consistently overrun its original budget and schedule by 30 percent, the estimate for the remaining activities could be increased by 30 percent.

Few of the popular parametric models work well when applied to operations and maintenance projects. The problem begins with size. In many cases, neither function points nor lines of code accurately reflect the workload of a maintenance project. Operations and maintenance projects often emphasize change requests and problem reports, and the technical activities tend to be organized differently than development projects. Simple estimating relationships based on local data can provide good results. For example, estimates of operations and maintenance resources may be based on the average effort per change from a similar project. Major enhancements (the other part of operations and maintenance) can be handled like development.

3.2.2 *Map and Calibrate*

Each estimation approach must be tailored to the unique characteristics of the project. This involves two major considerations: mapping the scope of the estimation method to the scope of the project, and calibrating the method with local historical data.

Most estimation methods make some assumptions about the activities and products included in the estimate. For example, many parametric models do not cover requirements analysis. Most assume that a minimal level of documentation will be produced. If the project being estimated includes a requirements analysis activity and is required to produce a substantial amount of documenting, the model needs to be adjusted. Be sure to map the development approach assumed by the estimation model to the development approach that will be used by the project.

Calibration is an important consideration. Analogy, activity-based estimating, and simple estimating relationships must use some historical data from the local organization. Parametric models generally are developed based on historical data from many organizations, none of which may be similar to the local organization. Some estimation tools provide a facility for entering local data and "running the model in reverse" to calibrate it. The historical data used to calibrate a model should be as similar as possible to the proposed project, including the application domain, process, and personnel.

Once the project is underway, data from early activities can be used to calibrate the estimation approach. For example, actual productivity from the first build can be used in a simple estimating relationship for planning subsequent builds.

No estimation method guarantees good estimates. Experience proves that model accuracy is specific to an organization. Using historical data from the organization is essential to the accuracy of estimates. There are several points to consider when using historical data to estimate or to calibrate a model:

- **Source of the data -** It is important to know the data's project characteristics, such as the duration of the project, the application domain, project problems (e.g., high staff turnover, change in funding level, excessive overtime), and when the project completed. Historical data from recent successful projects that are similar in application domain and duration to the current project is preferred.

- **Scope of the data -** Product and activity data can cover different contiguous phases of the life cycle (e.g., software and system requirements analysis, software and system architecture design, software detailed design, software coding and testing, software and system integration and test). Separate data for each phase is most useful because it can be organized to match the phases being estimated. If data is not available for each individual phase, adjustments must be made for the percentage differences between what the data covers and what is needed.

- **Level of detail -** Data is not always available at lower levels of product and activity detail. It is important to understand the aggregation structure of the data. While parts of past projects may be a close match to the project under estimation, the data may also include dissimilar pieces.

- **Effort attributes -** The number of staff hours (or days or months) can include different labor categories and different activities. It is important to know what is included in the effort data (e.g., labor for suppliers, testers, technical writers, managers, or administrative support). If the data is expressed in staff months, determine the number of hours in a staff month for data normalization.

- **Schedule attributes -** Schedule data is affected by the project phases and the activities in each phase. Individual phase dates and a list of phase activities provide the most flexibility for estimating the schedule requirements of new projects.

- **Size attributes -** Size data may include added, modified, or revised components. It is important to differentiate these attributes, because they affect effort and schedule differently. It is also important to know if the size data includes the deliverable product, test products, support materials, or a combination of these. For software data, consider the programming language used (lower-level languages consume more effort and schedule than higher-level languages).

- **Defect attributes -** When using historical effort and schedule data, it is important to know the quality of the product. Quality can be sacrificed for effort and schedule. Know what is included in the defect data (e.g., fixes, upgrades, enhancements). The number of defects discovered and the number resolved before project completion should be categorized by the severity of the defect.

Identifying projects within the same organization that are similar in size, application domain, anticipated staffing level, and anticipated schedule reduces many problems in using historical data. Improving estimate accuracy motivates many organizations to collect and store data on multiple projects in an organizational repository.

3.2.3 *Compute Estimates*

The estimation process produces numerical predictions for the size, effort, schedule, and quality of the project. These estimates depend on each other and usually are performed in the order listed. However, it is frequently necessary to iterate these tasks to achieve an estimate that satisfies all project constraints.

Size Estimation

Typically, the first element of an estimate is product size. The product size estimate may be prepared in terms of functional size, physical size, or both. Since it is the first activity in the estimation process, obtaining a good size estimate is essential to generating good estimates of effort, schedule, and quality. Product size is closely linked to the type of work required to produce the desired system. Consequently, size estimation for different types of work are handled differently. Types of software work include new development, COTS component integration, and maintenance. Unfortunately, few of the models and tools available accommodate the full range of common software scenarios.

The first step in estimating size is to select a size measure. Two size measures used in software estimation are lines of code and function points. Both of these measures have several popular variants.

The function points measure considers software from an *external user* perspective. It counts external connections such as inputs, outputs, interfaces, and reports. Function points are especially useful in user-driven systems such as transaction processing and information systems. Function points estimation requires knowledge of the system's functionality and of the function points counting methodology.

The lines of code measure considers software from an *internal structural* perspective. Counts of lines of code require knowledge of the potential software design. Source lines of code often works well with analogy and activity-based estimation methods. The rules for counting lines of code are simpler that those for function points.

Software size can be measured in many other ways such as classes, screens, or packages. Choose the size measure that best captures the work that is being performed.

- **Initial Estimates** - Estimating size early in the life cycle requires an examination of software requirements and specification data. Early in the life cycle, function points are counted directly from requirements and design specifications. Lines of code estimates are often based on an analogy between the current system and previous software designs. Alternatively, a high-level architecture may be developed and used to estimate the number of components. Components are then converted to lines of code by estimating the average lines of code per component (based on past experience). The best results are obtained by estimating size to the lowest degree of resolution that is economically possible. Typically, separate size estimates are produced for each software component.

- **Re-Estimates** - Product size estimates should be updated during the course of development. This may use the same initial estimation model, but better results are obtained by incorporating actual project data. For example, the number of components can be determined more accurately as soon as the design is completed. Thus, the actual number of components designed can be used instead of the estimated number to estimate size.

- **Non-Developed Item (NDI) Size Estimates** - A new system may include code from many sources, including new, modified, reused, and COTS software. The amount of code from each source should be estimated separately, since associated estimates differ from estimates of new development.

Effort Estimation

Effort estimation is widely understood and has the most support from estimation models and tools. During this activity, the product size estimate is converted into an estimate of effort by applying an appropriate estimating relationship, such as productivity.

Most of the effort associated with reused and COTS software comes from the need to integrate that software with developed code. The cost of integration can be significant. If any reused or COTS software must be modified as part of the project, effort will increase dramatically.

Cost usually is computed from the estimated effort using organization labor rates (using dollars per hour). COTS and reusable software may also have a licensing or purchase cost to include in the project estimate, even though it does not contribute to the effort estimate.

Schedule Estimation

Schedule estimation primarily answers the question: "How long will the project take?" Project duration can be estimated using the techniques discussed previously. A parametric model may be used for this purpose. The bottom-up or activity-based method estimates the calendar time required to perform each constituent activity. This method can be used to develop detailed plans for the project. However, a parametric model should also be used to provide a sanity check on a bottom-up or activity-based estimate. Most parametric models provide an estimate of project duration as a function of the estimated effort required for the project.

One of the primary mistakes made when estimating the development schedule is to ignore critical path dependencies between components. There are limits to parallel development and schedule compression. What may seem logical from a mathematical perspective may not be feasible in engineering terms.

Another important question is: "What is the best staffing profile (in terms of applied effort per month) over the project duration?" Several mathematical models exist for estimating optimum effort distribution. These models generally produce a curve with a single peak, indicating a gradual build-up and a gradual decline of resources throughout the project. This model form is especially valuable during the initial stages of project planning to determine the peak level of personnel resources needed. See Figure 4-15 for an example.

Of course, on short-duration projects, it is inconvenient to continually hire and place staff consistent with such a staffing profile. In addition, many organizations operate with a fixed staff level due to funding constraints. Make sure that the appropriate staffing profile is used in developing schedules.

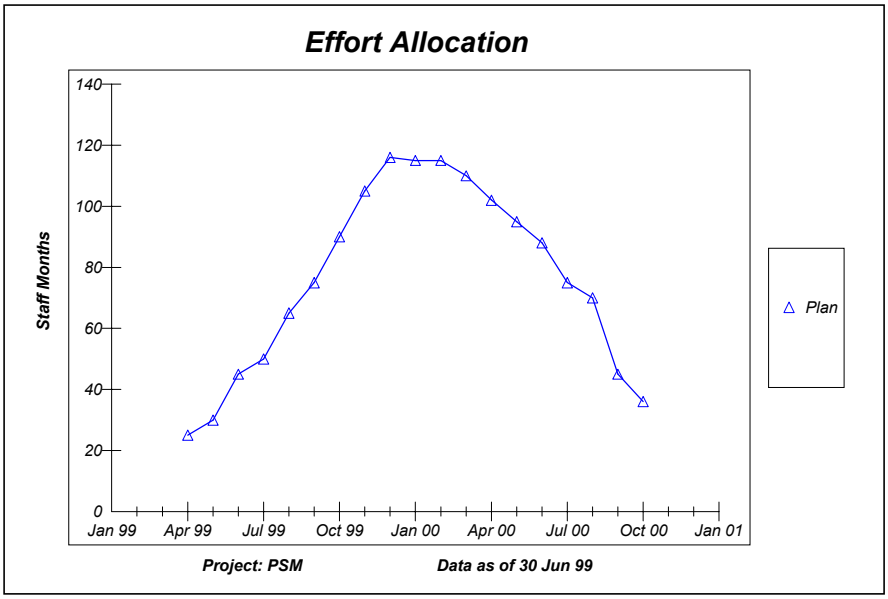

Figure 4-15. Effort Indicator Based on the Raleigh Curve

Quality Estimation

Systems often must meet specified quality objectives. For example, on some projects all known Priority 1 failures must be corrected prior to system delivery. Alternatively, reliability may be specified in terms of a minimum mean-time-to-failure. In order to achieve such quality objectives, the likely defect (or problem) content of the software must be estimated and actual defect trends must be tracked.

Use defect rates from past projects to make initial estimates of quality. Estimating defects from ongoing project performance is more difficult. Models that support this type of quality estimation fall into two categories:

- **Reliability models** - measure mean-time-to-failure, usually during integration and test
- **Transaction models** - measure defect insertion and detection rates throughout the life cycle

Both models require systematic data collection over multiple projects to develop good estimates for a specific project.

Quality objectives may be set in any of the measurement categories under *product quality*. Efficiency targets are usually derived from system performance concerns. For example, memory and CPU utilization are driven by the system response time requirements.

Quality objectives may also be set in terms of portability, usability, and maintainability. Setting these targets involves expert judgement since few organizations have adequate historical data and software requirements cannot be derived easily from system-level requirements.

3.2.4 *Evaluate Estimate*

The estimates should be evaluated from three perspectives:

- **Satisfaction of constraints -** Projects often are constrained by overall cost or required delivery dates. Even a high quality estimate may yield results outside those limits. If estimates resulting from the preceding tasks overrun constraints, make appropriate adjustments. When adjusting estimates, make trade-offs. For example, reducing functionality may be an effective strategy for reducing costs to meet a project constraint. Be sure to document adjustments and rationales.
- **Documentation of the estimate -** Good documentation improves the estimation process and facilitates periodic re-estimation. During re-estimation, determine if initial assumptions have changed. Information about the estimation process identifies potential areas for improvement.
- **Quality of the estimate -** Key considerations in determining the quality of an estimate include:
  - Has a firm foundation been established for the estimate of product size?
  - How well does the life cycle assumed by the estimation model map to the project's process?
  - Have all significant parts been included?
  - Has the estimation model been calibrated with local historical data or recent project performance data?
  - Have reasonable assumptions been made about performance factors affecting productivity, schedule, and quality?
  - Are aggressive goals or targets supported by realistic strategies for achieving them?
  - Are the results of alternative estimation methods consistent?
  - Has the level of uncertainty for inputs and outputs of the estimation process been identified?
  - Have estimating relationships been adjusted so that the results meet pre-defined project constraints?

Estimates that do not satisfy these evaluation criteria should be reconsidered. Poor estimates reduce the likelihood of project success.

Other non-software costs and schedules must be considered in the evaluation of project estimates. The costs of hardware and software purchases must be combined with development costs to evaluate project cost.

3.3 *Feasibility Analysis*

Feasibility Analysis evaluates the realism of plans associated with an issue. Estimation produces an initial plan; *Feasibility Analysis* either confirms the plan or produces alternatives. For a project plan to be feasible, the individual elements of the plan must be technically realistic, achievable, and consistent with each other. For example, Figure 4-16 shows a milestone (Gantt) chart and staffing profile for a project. Note the highly parallel design and implementation schedule between June and November during an interval of decreasing staffing. While the overall schedule may be adequate and the overall staffing may be sufficient, the allocation of staffing over time does not match the schedule. If this plan is followed, the project will experience some periods of overstaffing and understaffing.

Usually, only parts of an overall plan are unrealistic. Recognize and correct those situations to ensure project success. *Feasibility Analyses* should be performed throughout the software life cycle as plans are developed and revised. Projects fail when plans are unrealistic.

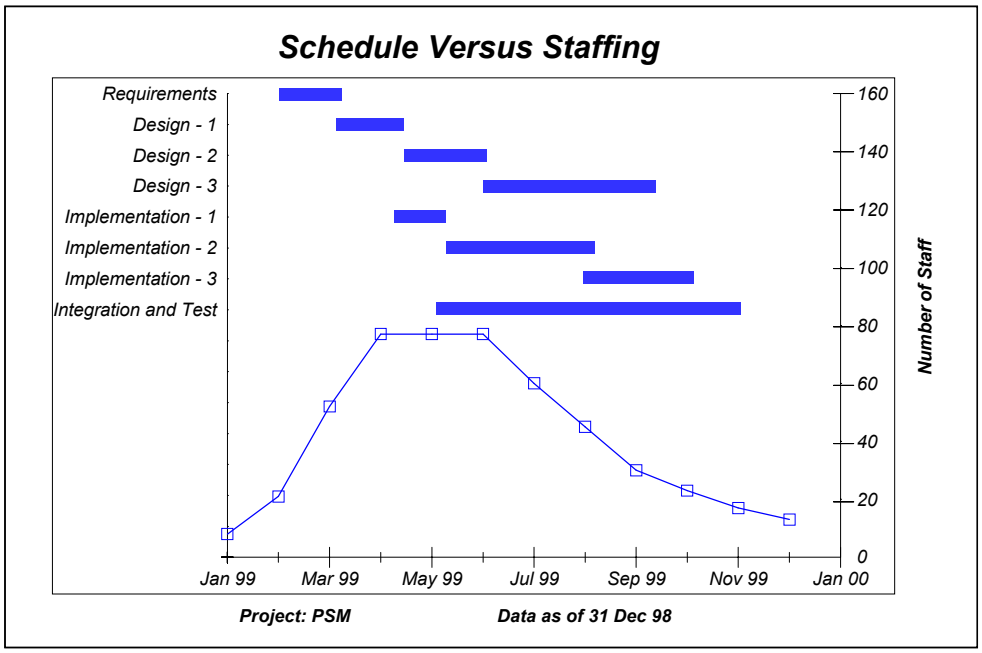

Figure 4-16. Project Schedule and Personnel

Plans often fail because they provide insufficient detail to effectively coordinate project activities. Most projects develop high-level estimates of size, effort, cost, and schedule, but often fail to allocate the estimates to lower-level project components and activities. Without these detailed plans, status cannot be evaluated objectively. Moreover, the lack of detail makes it easier for important project elements to be omitted from the

plan. If not discovered until late in the project, these omissions could seriously impact project success. Thus, the first activity in analyzing the feasibility of plans is to check that they are complete.

Feasibility Analysis should be performed using information about individual plan elements (such as budget and schedule) and by integrating information from several plan elements. The following sections explain how to assess different types of plans for feasibility in a systematic and integrated manner.

Each individual plan element that is related to a high-priority project issue should be evaluated against the criteria provided in Section 3.2.4. Then, related plan elements are compared for consistency, using the structured analysis model. For example, size estimates should be compared to the planned effort, and planned effort should be compared to scheduled activities. The feasibility of a plan depends on the accuracy of historical data and estimates, and on the effectiveness of the planning process.

Project development plans should be evaluated for both *breadth* and *depth*. The depth of a plan focuses on a detailed feasibility and internal consistency check of each plan element (such as schedule). For example, the measurement analyst should check that aggregated totals and summary values match more detailed counts and values, and that timeframes match on both high- and low-level plans.

Consider the following items to ensure that plans are defined in sufficient detail:

- **Resources -** organizations, people, and computers
- **Activities -** life-cycle tasks, support tasks, holidays, and vacations
- **Components -** units, test cases, lines of code, requirements, documentation, circuits, modules, and subsystem assemblies

Evaluating the breadth of a plan focuses on consistency and compatibility across plan elements. For example, the measurement analyst should check that totals of the same planning element (such as software components) match across plans, and that effort allocation peaks and valleys correspond to scheduled high and low activity levels. Breadth is particularly important for recognizing an incompatible element that can adversely impact related plan elements.

Follow these guidelines to assess the feasibility of specific plan elements:

- The overlap between project activities should be reasonable across the schedule.
- The rate of planned progress (slope on the indicator) should be reasonable.
- Planned performance should be consistent with past performance.
- Targets such as complexity and defect density should be reasonable within the project context.

Compare a plan to prior plans or actual performance data to assess the soundness of a plan. If a replan adjusts a plan without corrective action to solve underlying problems, the new plan will be just as impractical as the old one. Since objectives for portability, maintainability, and usability are usually derived subjectively, examining actual quality results from past projects is the best way to determine whether or not a quality objective is achievable.

Feasibility analyses should be conducted during initial planning and whenever significant changes are made to plans. Significant changes that warrant analysis include:

- The scope of work has changed due to functionality or requirement increases or decreases.

- Organizational or technological assumptions have changed. Examples include replanning that occurs when new technology has proven infeasible, or when organizational resources are not available as expected.

- *Performance Analysis* indicates that current plans are not being met.

Once a project begins, plans may change to incorporate new project information. A lack of re-planning may suggest that the project is not being actively managed towards its objectives.

Relationship of Plans

As discussed in Section 3.1.2, project issues are not independent. The structured analysis model can serve as a *Feasibility Analysis* "roadmap." The model can identify plan elements that are subject to *Feasibility Analysis*, as well as supporting consistency checks between plan elements. Three types of situations should be examined during *Feasibility Analysis*:

- Examine each individual issue (such as schedule) to check the plans for consistency. The analyst should review the schedule at a high level (such as milestone schedules) and at a low level (such as work unit progress) of detail.

- Plan elements that are upstream from the high-priority issues should be assessed individually because their validity will influence the feasibility of downstream plans. For example, if a primary concern is schedule, and the schedule plan looks feasible, effort should also be reviewed. Schedule could still be jeopardized because the amount of planned effort may be unrealistic.

- Plan elements should be assessed for consistency with other project information. For example, even if the individual elements of schedule and effort allocation are feasible, they may not be synchronized with staff availability. These situations must be recognized and resolved to maximize project success.

Feasibility Indicators

An indicator is a measure or combination of measures that provide insight into a software issue or concept. An indicator usually involves a comparison. During Feasibility Analysis, the comparison may be between sets of plans, or between plans and historical data. Two types of indicators may be defined: those based on trends, and those incorporating thresholds or targets. The following paragraphs provide examples and evaluation guidelines for each type.

Sometimes plans are represented as trends over time. Trends are usually represented in one of two ways: as cumulative plans or as profile plans. A cumulative plan shows the total quantity planned to be achieved to date, such as the total cost to date. A profile plan shows the planned quantity apportioned to each reporting period, such as the number of staff assigned to the project each month. Figure 4-17 is an example of a cumulative plan, and Figure 4-18 is an example of a profile plan.

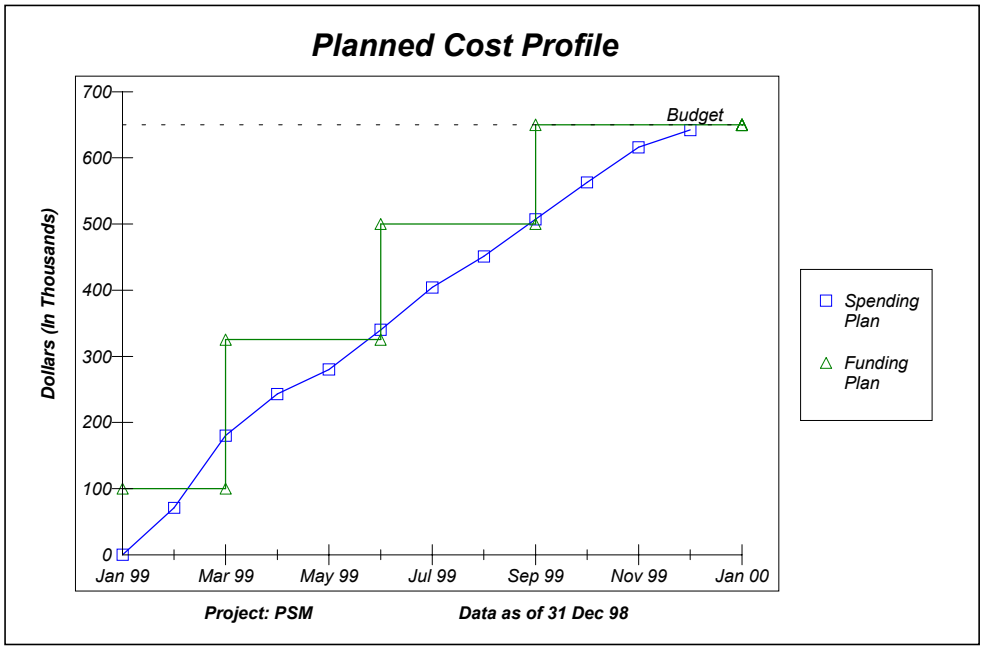

Figure 4-17. Cumulative Plan Example

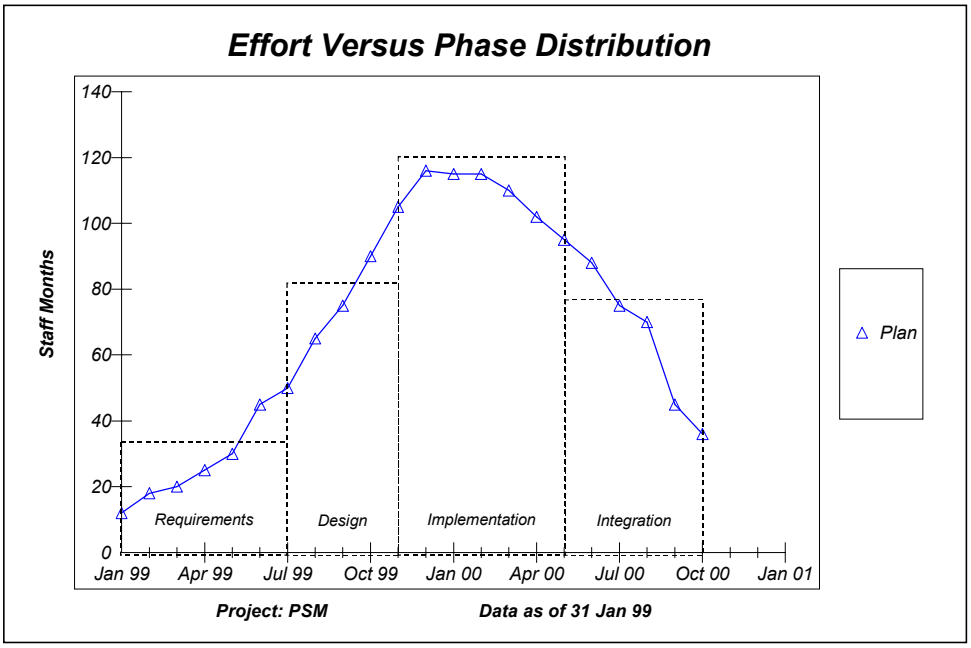

Figure 4-18. Planned Effort Profile Example

Sometimes, a planning baseline is really a single value - such as target, goal, limit, or threshold - against which a set of actual values will be compared. Many of these targets originate with product requirements, such as code complexity, response time, or memory utilization. They may also be the basis for final software acceptance criteria. These types of baselines are typically represented as a straight line on a graph. For

example, figure 4-19 shows how data from previous systems can be used to assess the realism of a response time limit. Notice that several past systems met or almost met the limit; therefore, the baseline seems feasible.

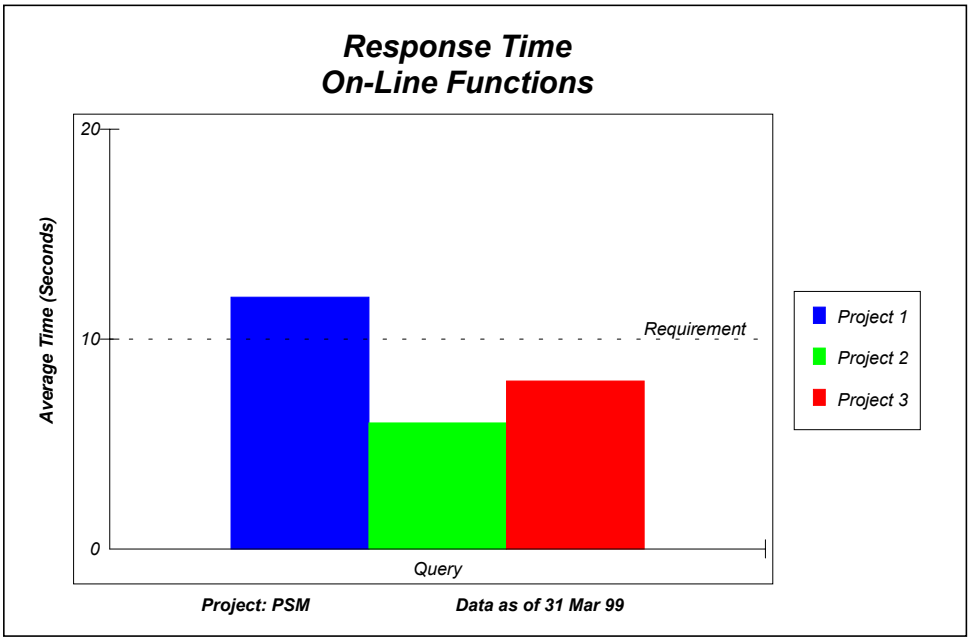

Figure 4-19. Threshold Baseline Example

Measurement goals are sometimes used to derive the measurement baseline for an indicator. For example, in operations and maintenance, a goal might be that the backlog of change requests should not exceed some defined number. Another example is that no more than a certain percentage of discovered defects should remain open at any point in time. These goals can be compared to the actual backlog and actual percentage of open problems during *Performance Analysis*.

Evaluating Targets

When conducting Feasibility Analysis of targets and thresholds, consider the overall reasonableness of the target value as well as the potential for specific problems. Common concerns are:

- Does the target or threshold appear reasonable given the technologies used, the project management strategies, and the schedules and activities?
- Does a sound precedent indicate that the target level of performance is possible, based on other similar projects, new technology applied, or other factors?
- Does the target support overall project objectives?
- Have specific risks or obstacles that might affect achievement of the target have been identified?
- Is the target outside the confidence limits established by historical data?

The distance between the selected target and the point estimate can be expressed as a standard deviation. Confidence limits are often set at plus or minus two standard deviations. The chance of achieving a result outside these confidence limits is less than five percent. Plans based on targets selected beyond the deviations are suspect.

Complicating Factors

Plan elements often conflict because different people develop them at different times, using different strategies. For example, a technical manager may use one strategy for creating a detailed work unit delivery plan, while the project manager may use a different strategy for cost and effort allocation. Feasibility Analysis is complicated by the fact that multiple plans must be assessed:

- High-level estimates and plans are created during the project-planning phase. These are typically based on mission requirements and general assumptions about the development approach. These early plans should be assessed for feasibility.

- More detailed project plans are generated in response to allocated requirements. In contracting situations, these estimates and plans may be included in the proposal. Their feasibility may be a factor in source selection.

The planning process can run into problems when too many participants are involved. Detailed project plans are often developed by individuals who did not create the original estimates. Assumptions may not be carried over to the detailed planning process, or new assumptions render the original estimates infeasible. The latter situation often occurs when estimates are elevated through multiple levels of management approval. During a Feasibility Analysis, the measurement analyst must ensure that plans are based on sound engineering judgment or historical data, rather than on unrealistic constraints.

In order to assess the impact of a planning problem, the source of the problem must be localized, and its scope must be evaluated. This may require an examination of both the planning assumptions and the basis for the estimates. The estimation techniques discussed in section 3.2 can help assess the impact of a planning problem. The outcomes for both the best and worst case situations should be assessed.

Always identify alternative courses of action for infeasible plans, including their approaches, assumptions, and constraints. For example, if a project's scheduled delivery date is infeasible, the following alternatives might be considered:

- Reducing requirements

- Increasing staffing

- Adding more experienced staff

- Extending the delivery date

- Using additional COTS components

Some alternatives may not be viable, given project constraints and others may add risk. Always inform the project manager of all feasible alternatives.

3.4 *Performance Analysis*

Performance Analysis determines if the project is meeting defined plans and targets. Regardless of its feasibility, once a project has committed to a plan, performance can be measured against the plan. Even if the project begins with a good plan, once performance begins to deviate from the plan, the reasons for the deviation must be identified and corrective actions taken to ensure success. The goal of *Performance Analysis* is to provide information for making decisions in time to affect the project outcome.

The following discussion illustrates the PSM approach to *Performance Analysis*. Figures 4-20 and 4-21 show a problem discovered through analysis of multiple indicators. Figure 4-20 presents a design progress indicator. While actual design progress appears to be only slightly behind the plan, the number of open problem reports generated during design inspections (Figure 4-21) continues to increase. These open problem reports represent rework that must be completed before the design activity can be completed.

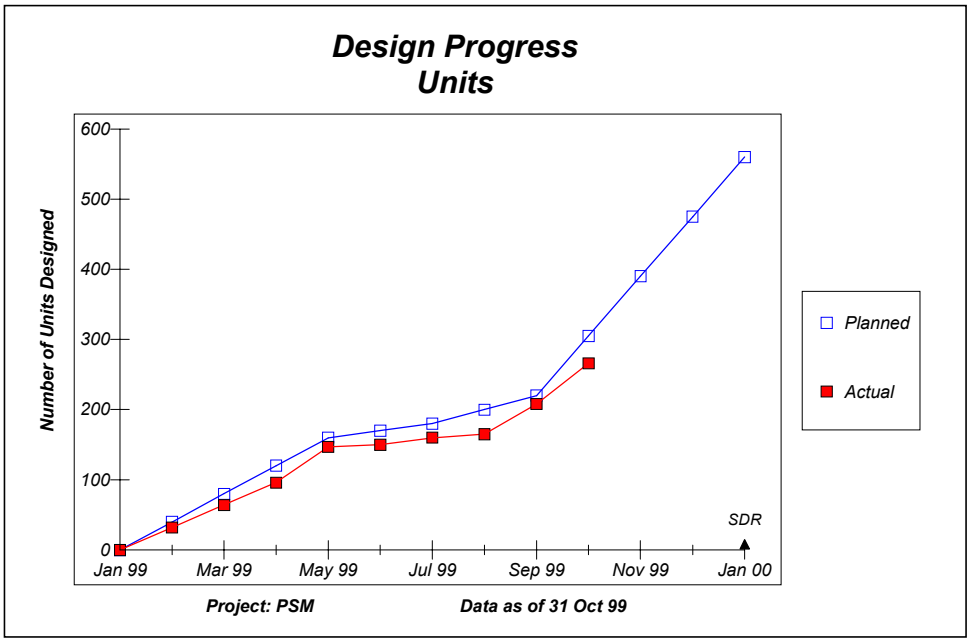

Figure 4-20. Design Progress Indicator

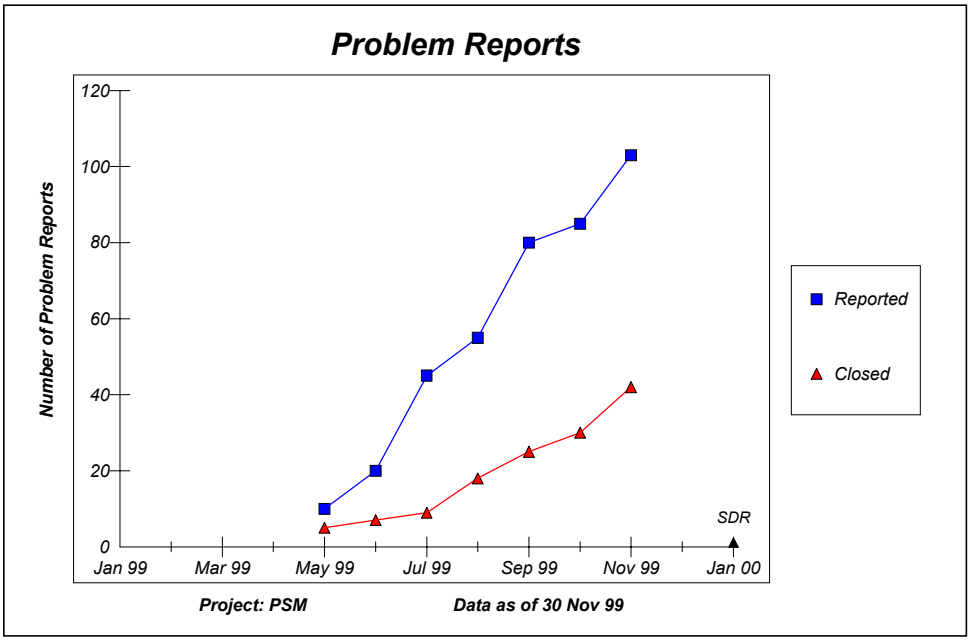

Figure 4-21. Problem Report Status Indicator

Once a problem has been identified, it should be localized by examining indicators based on more detailed data. Identifying the specific source of the problem helps to determine its cause and to select corrective actions. Figure 4-22 shows open problems by configuration item. It indicates that most open problems are associated with configuration item F.

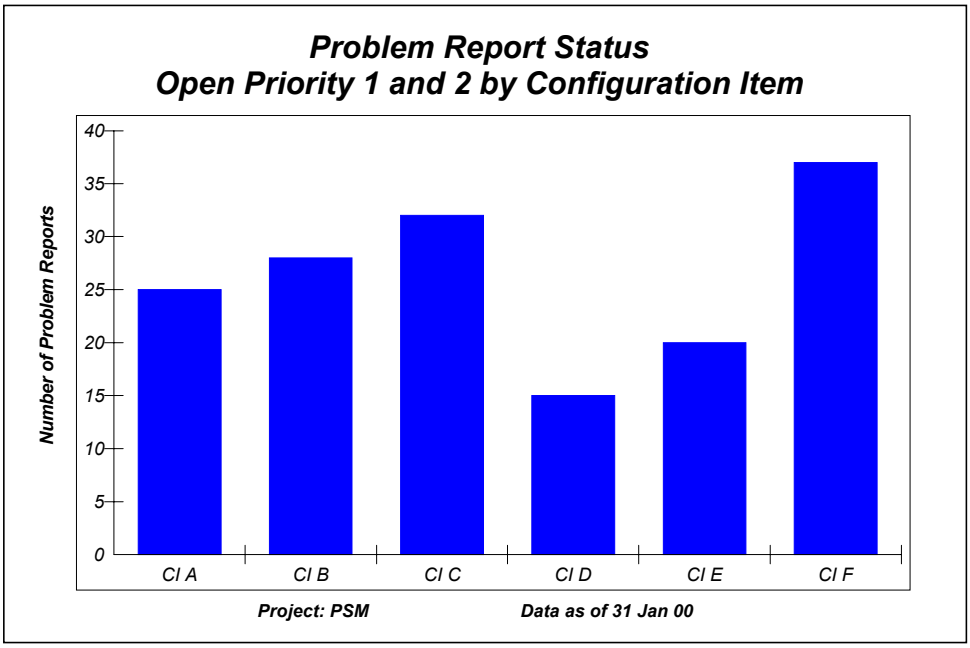

Figure 4-22. Open Problems by Priority

The guidance in this chapter does not define a rigid approach to *Performance Analysis*: the process must be flexible to analyze important project issues. *Performance Analysis* is an investigative process. It is necessary to look at the data in different ways and at different levels of detail in order to track and isolate problems. The following sections describe several analysis techniques and the process for using them to gain insight into project performance.

Performance Analysis consists of four steps, as shown in Figure 4-23. The first step evaluates actual performance against the plan. During this step, indicators are generated and analyzed. If problems or risks are identified, the next three steps must be executed. The second step is to assess problem impact by localizing the problem source(s) and evaluating the scope of the problem. Additional indicators may need to be generated. Next, the project outcome is predicted, usually by extrapolating from current trends in the data. Finally, if the predicted outcome does not meet project objectives, then alternative actions must be identified and evaluated. The resulting information is provided to the project manager for decision-making.

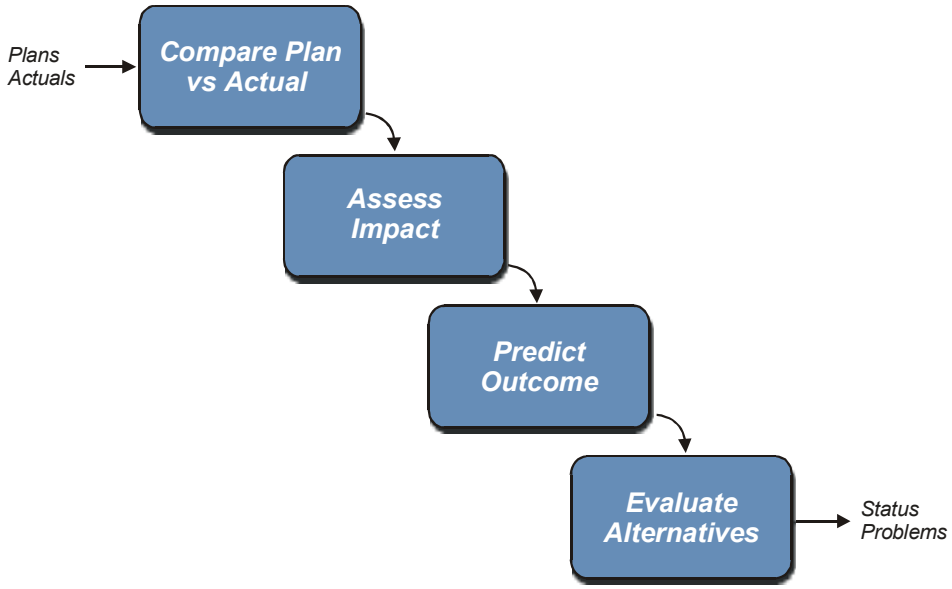

Figure 4-23. *Performance Analysis* Process

Several of these *Performance Analysis* tasks require non-measurement information. Decisions cannot be based solely on quantitative data. Project context information may be collected from project team members, joint technical and management reviews, document reviews, and risk analyses. Gathering and integrating non-quantitative information is essential to the successful application of measurement.

3.4.1 *Compare Plan versus Actual*

When evaluating performance, the basic indicators that correspond to each issue are examined. Problems are identified by quantifying the difference between plans and actuals. If the difference exceeds the threshold acceptable to management, then the situation should be investigated further. Both the trend and the absolute magnitude of the difference should be analyzed. If a variance has been growing steadily over time, it should be investigated, even if it hasn't exceeded a pre-defined threshold.

Because project issues are not independent, an *integrated analysis* using multiple indicators also must be performed. For example, a problem in one issue area (such as increases in effort) may be disguised by an accommodation in another issue area (such as schedule).

Using the Analysis Model

The PSM structured analysis model (discussed in section 3.1.2) is a useful "roadmap," with several implications for *Performance Analysis*:

- Indicators monitor the performance of high priority issues. By the time a problem is identified from a direct indicator, it has probably become significant.
- Always monitor elements that are upstream from the primary issues because they represent leading indicators for those issues. For example, requirements changes usually precede size and effort increases. If the primary concern is staying within a fixed cost, effort should be measured along with product size, process performance, and technical adequacy.
- Even if high-level indicators suggest the project is moving ahead smoothly, delays and quality problems in a critical path item (that are not recognized and countered early) can have a ripple effect late in the project. A good example is the need for hardware to begin testing.
- When evaluating alternatives, always weigh the trade-offs between issues. Optimizing a primary issue may negatively impact another issue that is equally important to project success. For example, attempting to make up a schedule slippage by increasing the number of personnel usually increases overall cost.
- Sometimes two related indicators suggest different situations. Neither variance alone may be large enough to suggest a problem, but taken together, they indicate that an element is not working as planned.
- Individual points that are not part of a trend, but that show unusual behavior, should also be investigated. Examples include a large change in productivity, defect rate, or complexity.

Performance Indicators

Performance indicators are classified into two general types, based on how they are graphed and analyzed: those based on trends, and those based on thresholds or targets. The primary distinction between the two is whether the expectation (plan or target) is relatively constant or changes over time. The following paragraphs explain these indicator types in more detail.

Trend-based performance indicators are used when the expected or planned value changes regularly over time. Figure 4-24 shows a trend-based indicator where a different goal or target for *work units completed* has been set for each week. This is the project's implementation plan. Actuals also are plotted on the same graph. *Performance Analysis* of a trend-based indicator determines if the actual project trend corresponds to the baseline or expected trend. Notice that the number of actual units completed in a given month is substantially lower than planned.

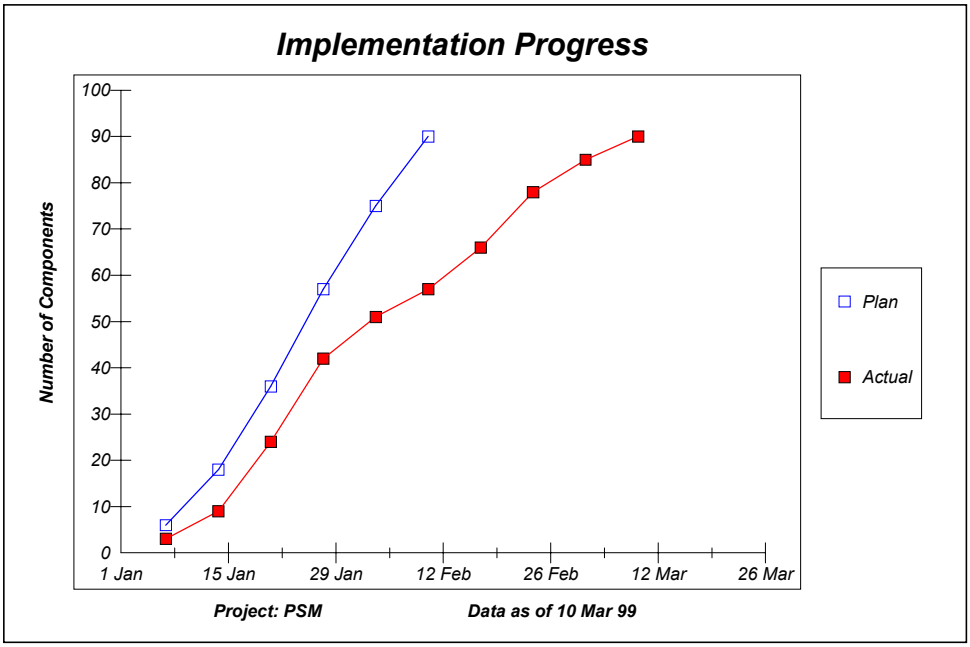

Figure 4-24. Trend-Based Indicator Example

Indicator baselines can also be totals that are not known in advance. For example, an alternative trend-based indicator tracks work backlogs for items such as problem reports. The amount of work to be completed, represented by the number of open problem reports, must be developed week by week as problems are discovered.

Performance indicators based on thresholds or targets are used when the expected value remains relatively constant over time. *Performance Analysis* determines whether the actual project performance meets or exceeds its established bounds.

Figure 4-25 is an example of a limit-based indicator for response time. As long as the actual response time remains within the planned limit (which may be a contract requirement), performance is acceptable. Whenever actual values exceed the limit(s), the cause should be investigated. In this example, response time was exceeded for on-line functions, but test results from the last two builds indicate that the problems have been corrected.

Limits can represent norms, expected values, or constraints. In many cases, limits are specified as requirements. In other cases, they represent threshold values established by the project manager. Limit-based indicators are often used to represent defect rates, complexity thresholds, computer utilization targets, and productivity goals.

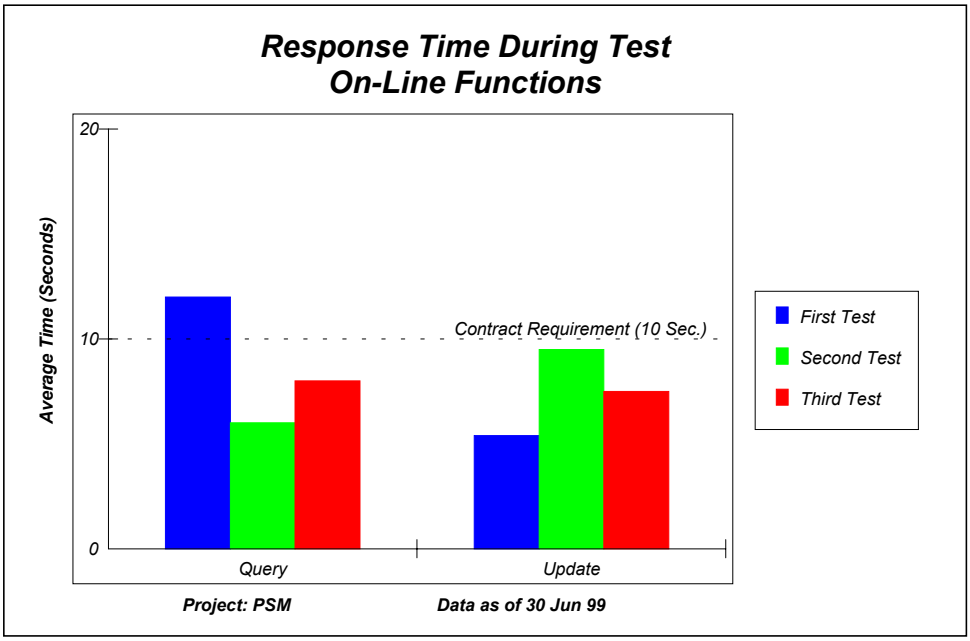

Figure 4-25. Limit-Based Indicator Example

A common technique for monitoring performance is to set thresholds around the planned performance. As long as the project progresses according to plan and within the threshold, no management action is necessary. Thresholds are displayed visually as boundaries around trend lines. Crossing a threshold may be interpreted three ways:

- There may be an unusual individual result. This unique occurrence may not require any immediate action. The situation may correct itself. However, that can only be determined once the reason behind the deviation is understood.

- There may be a constant variance of actual performance from the planned baseline. In this case, performance is outside the threshold, but it parallels the planned baseline. This may be due to a one-time perturbation in the project. Managers must either compensate for the deviation or accept the deviation and replan accordingly.

- There may be an increasing variance between actual performance and the planned baseline. This situation usually is serious and warrants immediate management action. The deviation may be caused by systemic problems that inhibit projected performance, or estimation errors that make the plans infeasible.

Thresholds are visual cues to alert management to investigate and possibly take action. Falling below a threshold indicates that work will not finish on schedule unless requirements or processes change.

There are three approaches for setting thresholds:

- **Arbitrary -** This is usually a percentage based on engineering judgment. It may not reflect a real need to take action when the threshold is crossed. In fact, the threshold may be ignored if it is not useful. This is the least desirable but easiest threshold to set. This type of threshold is represented as a line (or limits) parallel to the trend line.

- **Management reserve -** A reserve is the amount of unallocated resources that can be used for remaining work. A management reserve may be planned, such as a schedule reserve built into the plans. Another type of management reserve may not be explicitly planned, such as overtime effort. The threshold must be recomputed when any of the management reserve is consumed. This type of threshold typically narrows down over time as the management reserve is consumed.

- **Statistical limits -** With this approach, actual data is used to calculate statistical thresholds. These may be limits or confidence intervals that represent standard deviations (or sigmas) from the average or expected performance. They are the hardest thresholds to set because of the requirement for good historical data. The attraction of this approach is its objectivity.

Additional context information is usually needed to make valid interpretations about the cause of a problem. For example, noting a discrepancy between the original size estimates and the current (or actual) size estimates does not provide enough information for management action. The size difference may result from: 1) poor initial estimates of system size, 2) significant requirement changes, or 3) changes in the way size is counted. Depending on the cause of the variance, different actions may be required.

In many cases, measurement results can be compared and evaluated within the boundaries of the project to identify problems. This is especially true for large projects with many activities and components. For example, if a project has a large number of configuration items, defect densities of the configuration items with similar designs and functions can be compared to identify "outliers" (those components with an unusually high number of defects).

Figure 4-26 shows an example of this type of analysis. The Navigation CI experienced the highest defect rate, but also demonstrated the highest complexity. This analysis suggests that extra attention should be placed on this CI, perhaps in the form of increased levels of inspection or testing.

Software Quality

CI	Size (KSLOC)	Total Valid Defects	Defect Density	Number of Units	Average Complexity	Units with Complexity >10
Navagation	124.8	42	0.34	156	9.2	26
Sonar	45.6	12	0.26	68	6.7	15
Weapons	56.6	14	0.25	75	5.2	8
System Services	75.3	20	0.27	102	7.5	16
Display Services	168.0	32	0.19	125	8.6	12
Training	25.5	3	0.12	42	4.2	3
Total / Average	495.8	123	0.25	568	6.9	80

Project: PSM *Data as of 31 March 98*

Figure 4-26. Sample Matrix of Measurement Results

Sometimes inconsistent, incorrect, or inaccurate data may cause an indicator to suggest a problem where none exists. All data anomalies and other potential inconsistencies should be reviewed with the data provider. However, when multiple indicators point to a problem, it is usually not just a data issue.

3.4.2 *Assess Impact*

The first task in assessing the impact of a performance problem is to localize the source of the detected variance and to evaluate its scope. This may require additional focused data collection, but usually can be satisfied with existing data.

Sometimes a substantial difference between planned and actual values may be caused by outliers, which are values that do not appear to be consistent with other data. For example, the average cyclomatic complexity of a component may be significantly higher than others in the system because of one or two unusually complex units. Judgments about the whole system should not be based on these outliers.

Once the source and scope of the problem have been identified, its potential impact on project success can be assessed. The magnitude of the impact is not always proportional to the difference between planned and actual values. Sometimes, a small problem that arises in one issue area may have a ripple effect on another issue, multiplying its effect.

3.4.3 *Predict Outcome*

In order to fully appreciate the significance of a problem, its impact must be projected into the future. Project outcome can be predicted by projecting current trends as straight lines (for measures such as work unit progress, size growth, and requirements changes) or by employing more sophisticated parametric estimation models (for measures related to effort, size, schedule, and problem reports). See Section 3.2 for details.

Alternatively, the variance to date can predict future performance so that plans can be adjusted. For example, if progress to date has been twenty percent below plan, future progress may also be assumed to lag by twenty percent, *unless some specific action is taken to change the project's performance*.

Use these projection techniques to evaluate the effects of changes on project outcomes. Exploring "what-if" scenarios helps in understanding which factors most strongly influence project outcomes. They also help to determine whether a project can make up the gap between planned and actual performance. Throughout these studies, keep in mind the imprecise nature of such projections. Small differences in predicted outcomes are probably meaningless.

3.4.4 *Evaluate Alternatives*

If the predicted outcome does not satisfy project objectives, alternative courses of action must be investigated. Measurement assists in evaluating project outcomes given different scenarios and actions. Historical data and qualitative experience from similar projects also help in evaluating alternatives.

Use the structured analysis model to identify where trade-offs can be made to bring the project plan into line with project objectives. For example, if the projected cost from the preceding task exceeds the project constraint, look for upstream issues that may influence and adjust cost with appropriate action. Eliminating requirements might reduce the amount of work to be done. If the current schedule is aggressive, some benefit from reduced rework may be obtained by extending the schedule. Increasing the level of automation might enhance productivity.

Consider the effect of each alternative on the risk and financial status of the project, as well as on the current problem. An action that addresses a current problem could increase the risk exposure of the project in other ways. For example, purchasing and implementing a productivity aid such as a design tool might lower costs in the future, but the risk of delays in acquiring, installing, and learning the tool might make this course of

action undesirable in the near term. Focusing on current problems can force a project into a situation where it cannot recover from a significant risk.

The viability of each proposed course of action should also be weighed against financial performance. The budget and schedule may affect the implementation of a proposed action. All these sources of information help the project manager arrive at an optimum decision within the bounds of project constraints.

Keep in mind that the purpose of this task is to identify specific actions that can be taken to change the outcome of the project. Just changing the assumptions behind the project plan does not make the plan any more viable. For example, assuming increased productivity without taking any real action will not prevent cost overruns.

Underlying problems and potential actions should be reviewed with the project team and modified as appropriate. Consider the data, the performance indicators, and context information about the project and recent events. Do not make conclusions based on a single item, whether quantitative or subjective. In deciding on a specific recommendation, consider the nature and effectiveness (or impact) of previous corrective actions.

Once a likely alternative has been identified, it needs to be passed back to the *Predict Outcome* task. In order to make a decision, the project manager needs to know what alternatives are available, as well as the likely consequences of each alternative. One result of the analysis process may be to identify a new issue and to recommend the collection of additional data to track it. This may require revisiting the *Tailor Measures* activity described in Part 2 of the Guide.

[This page intentionally left blank.]

4

Make Recommendations

The purpose of measurement is to help project managers make better decisions. The final task in the PSM *Apply Measures* activity involves reporting analysis results along with recommendations. The analysis task identified and evaluated alternatives. Now these need to be presented to the decision maker. The decision maker will then select from alternative courses of action and implement corrective actions based on the available information.

Information from the *Analyze Issues* task is input to the *Make Recommendations* task. This analysis information includes a "status report" on the currently monitored project issues, as well as on any new risks or problems derived from analysis. The information also includes alternatives for dealing with identified risks and problems.

Analysis results also provide important inputs to the project risk and financial management processes. Newly identified risks should be fed into the risk management process so that they can be formally recorded, tracked, and monitored. Measurement data helps monitor the probability of occurrence and likely impact of a defined risk. If the measurement and financial performance plans are coordinated during the *Tailor Measures* activity, measurement data also can be used to objectively assess the financial status of a project.

Once project status is understood, and risks, problems, and alternative courses of action have been discussed, the optimum alternative should be selected and acted upon. The responsibility for these activities lies outside the measurement process. The goal of the *Make Recommendations* task is to maximize the overall likelihood of project success. Thus, when selecting an alternative, consider the effects of the proposed action from the risk and financial performance perspectives, as well as from the measurement perspective. Finally, decisions are made and implemented with appropriate actions. Occasionally, available information may be insufficient to select a course of action. In this case, adjust the measurement process to collect the necessary data.

The following paragraphs discuss both the reporting and use of measurement results.

Report Results

Analysis results must be communicated regularly to the project manager and to the project team as a briefing, report, or on-line message. The reporting system should promote regular interaction and objective communication among all participants. Recognize that the measurement analysis report or briefing may contain proprietary or sensitive information. Therefore, the reporting methodology must include appropriate protection and security. The communication should include:

- **Overall evaluation of the project -** This includes status of known project issues and projections of performance through completion.
- **Identification of specific problems, risks, and lack of information -** The location, cause, and impact of current or potential obstacles to project success should be described, along with any noteworthy outliers or trends.
- **Recommendations -** This includes alternative actions, with advantages and disadvantages of each, to address the underlying risks and problems identified in the analysis.

- **Potential new issues -** The nature of the problem or proposed actions may result in *new issues* that may affect the focus of the measurement process.

Reporting and reviewing measurement results must be integrated into the day-to-day project technical and management processes. Results should be evaluated against project events, risk management results, and project financial performance. Measurement analysis results should also be input to periodic project status and milestone reviews. Product quality issues (e.g., reliability, maintainability, portability) are usually addressed at major milestone reviews rather than in monthly status reviews. Be sure to present results for all issues at one of these decision-making activities.

If possible, measurement results should be initially reviewed by the project team. This interaction may uncover events and qualitative information that help to explain the data. It is easy to arrive at incorrect conclusions without such communication. Measurement should be used for communicating and understanding, not for assigning blame.

Decision makers need to know how analysis results and recommendations were derived. All assumptions should be well defined. This helps managers justify decisions and trace recommendations back to the underlying data.

Analysis results should be used to update the project's risk management plan. Measurement of actual status and performance levels helps to re-assess the probability of occurrence and magnitude of risks. Quantitative progress measures also provide a solid basis for reporting and explaining financial performance, and can provide objective input to Earned Value or activity-based costing systems. All analysis information should be presented together to the decision maker as a basis for evaluating alternatives and taking action. Different decision makers have distinct preferences for the presentation of measurement results and alternative recommendations. Adjust reporting formats so that data and recommendations are clear.

Use Measurement Results

Using measurement on a project does not require any special or additional management functions. However, basic project management must be in place. Measurement complements existing planning and control activities especially risk management and financial performance management. When management action is appropriate (based on measurement and other project information) it should be implemented through the existing management structure and contractual mechanisms.

The measurement results present decision makers with alternative courses of action to correct or minimize a problem. The decision maker must choose the best alternative. One option is to do nothing. In most cases, a more proactive stance is necessary. Look for alternatives that address the underlying cause of the problem, not just the "critical" symptom. For example, if a project is running over its budget, try to understand the cause of the problem before adding money to the budget. Is productivity less than expected? Are the requirements expanding? If so, address these issues.

Assess the impact of each action on the project's overall risk status. An action that addresses a current problem could increase the risk exposure of the project in other ways. For example, purchasing a productivity aid, such as a design tool, might lower costs in the future. However, the risk of delays in acquiring, installing, and learning the tool might make this course of action undesirable. Focusing strictly on known problems can create additional risks.

Also consider the financial viability of each action. Budget and schedule constraints may eliminate a possible alternative. All of this information is required to help the project manager arrive at the optimum decision within project constraints.

Measurement helps to recognize and localize problems. Identifying a problem's cause and selecting an appropriate corrective action requires good management and engineering judgment. Action must be taken to realize any benefit from measurement.

Project managers may take several different actions, such as:

- Extending the project schedule to maintain quality
- Adding development resources to stay on schedule
- Deleting functional capabilities to control costs
- Changing the development approach or acquisition process to improve performance
- Reallocating project resources and budgets to support key activities

In an acquisition or outsourcing environment, some of these actions may have to be taken by the supplier instead of the acquirer.

Some of these actions affect project baselines and may not be taken unilaterally. Other actions attempt to optimize performance within the project's established constraints. Measurement, risk, and financial performance information help the project manager to recognize and select the best available course of action.

Once a corrective action is initiated, additional measurement indicators may be defined to assess its effectiveness. Normally, there is a delay between the start of a corrective action and its effects. Nevertheless, it is important to follow through to ensure that the desired outcome is realized. In most cases, new indicators can be defined using existing data.

Example of a Measurement-Based Decision

The following example describes a decision-making scenario from a real-time system with more than two million lines of code. The product had a rigid delivery date for integration into a military platform overseas. The system had to be highly reliable. The supplier proposed a two-build strategy for product delivery. The supplier expected to achieve high productivity in the first build and less in the second build. The second build would implement the more difficult functionality.

Figure 4-27 indicates a problem. The actual productivity to date shown in the figure was not as high as the proposed productivity. Continued performance at his level would jeopardize program success. Two alternative replanning options were proposed (see Figure 4-27). Choosing among the options required answering several questions:

- Had requirements changes affected the difficulty of the work?
- Was build content shifted so that more complex work was done first?
- Was project staffing insufficient?
- Was there a training period and learning curve?
- Was performance increasing over time during Build 1?

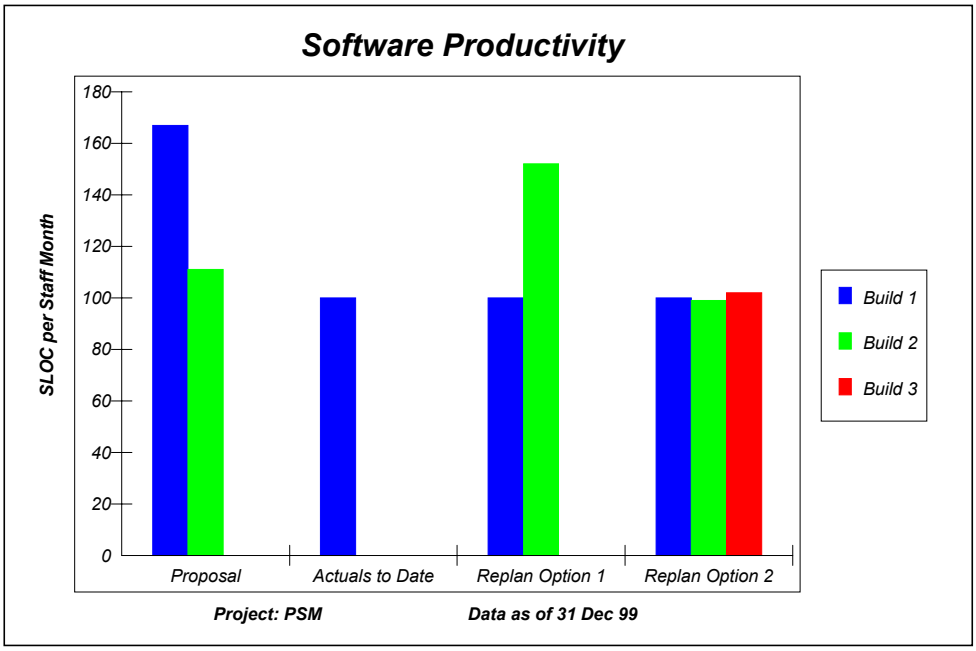

Figure 4-27. Supplier Productivity

Answering these questions would help to select between the alternatives. The supplier proposed to do a replan based on the original build strategy, but using a higher productivity level in Build 2. This proposal, called Option 1, was based on lessons learned in Build 1 that would increase productivity in Build 2. In Option 2, the acquirer proposed to add a third build with the assumption that Builds 2 and 3 would have similar productivity to Build 1. This proposal would be difficult to sell, since it required more funding and schedule. The product might achieve sufficient capability in Build 2, even for limited deployment.

One of the first investigations focused on the growth in requirements since the contract award. Many requirements were added between July 1997 and May 1998. After May 1998, requirements growth leveled off, as shown in Figure 4-28. However, changes to requirements had already affected the product size, as shown in Figure 4-29. Not only was the total amount of software larger, but also more of it had to be developed rather than reused. This argued against an increase in productivity in Build 2.

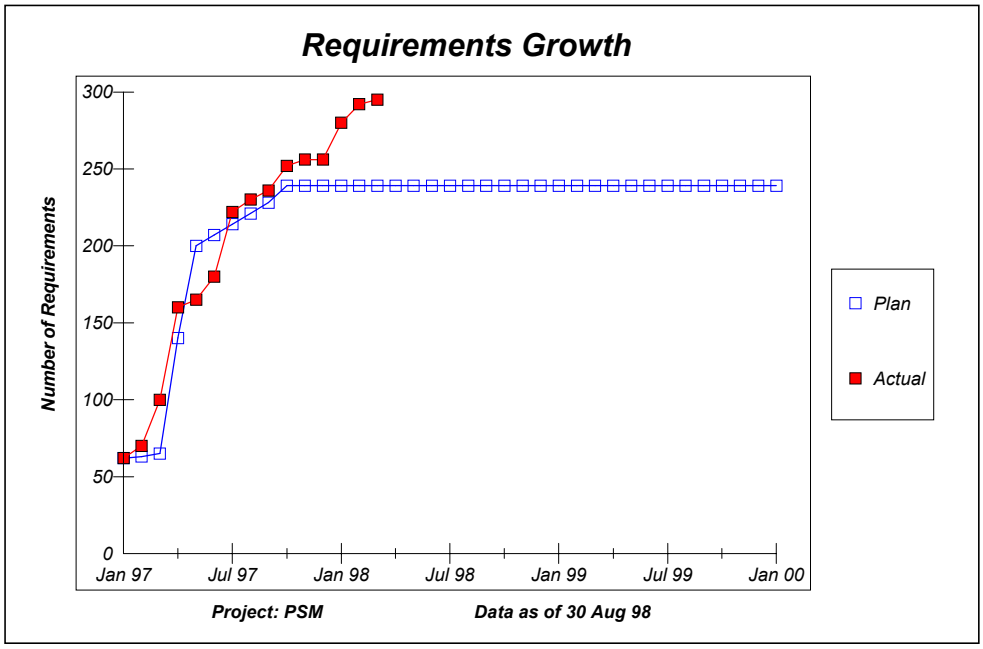

Figure 4-28. Requirements Growth

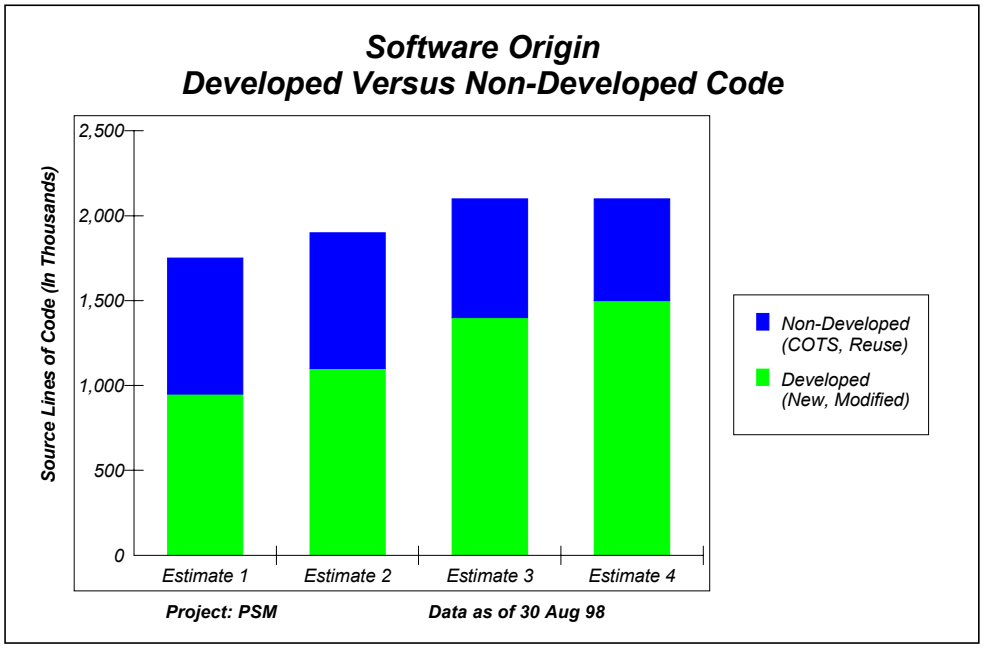

Figure 4-29. Software Size Estimates

The downstream effect of trying to do more work with fixed staffing and schedule usually is poor quality. Figure 4-30 shows the trend in resolution of problem reports. A substantial backlog of problems existed and was growing. The rework associated with resolving all of these problems would lower productivity on future builds.

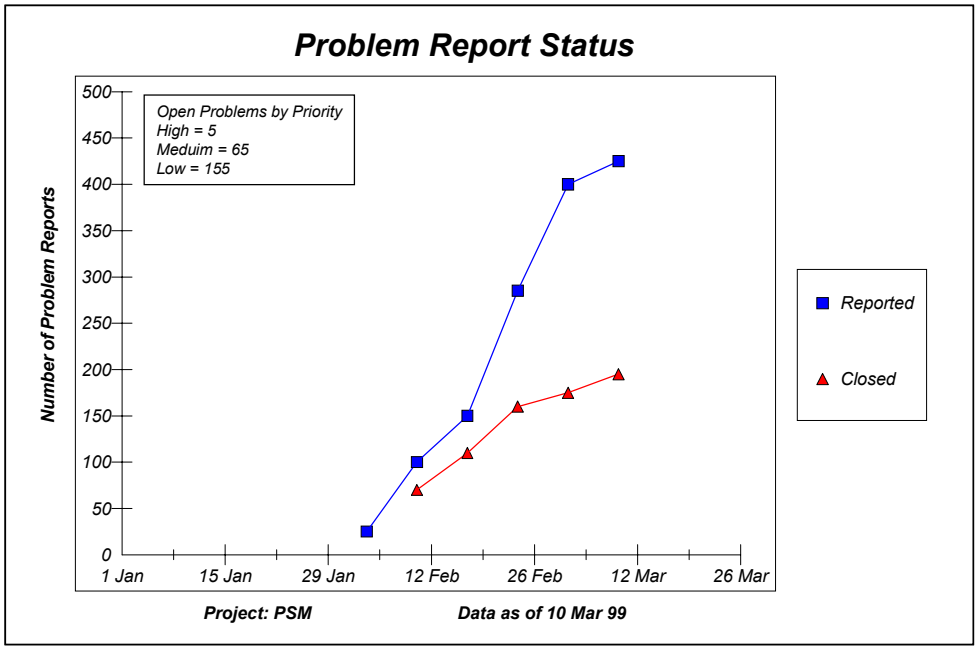

Figure 4-30. High-Priority Problem Report Status

Which replan option was the best? The evidence of a change in software source and increasing backlog of rework indicated that productivity increases would be difficult to achieve in Build 2. Only the option of adding a third build was feasible. The insights provided by these indicators helped to explain what was happening, so that the project manager could make an informed decision.